THESIS

REQUIREMENTS AND LIMITATIONS OF BOOST–PHASE BALLISTIC MISSILE INTERCEPT SYSTEMS

by

Kubilay Uzun

September 2004

Thesis Advisor:	Phillip E. Pace
Co–Advisor:	Murali Tummala

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY	**2. REPORT DATE** September 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Requirements and Limitations Of Boost–Phase Ballistic Missile Intercept Systems	**5. FUNDING NUMBERS**
6. AUTHOR(S) Kubilay Uzun	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center for Joint Services Electronic Warfare, Naval Postgraduate School, Monterey, CA 93943–5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Missile Defense Agency	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The objective of this thesis is to investigate the requirements and limitations of boost phase ballistic missile intercept systems that contain an interceptor and its guidance sensors (both radar and infrared). A three–dimensional computer model is developed for a multi–stage target with a boost phase acceleration profile that depends on total mass, propellant mass and the specific impulse in the gravity field. The radar cross–section and infrared radiation of the target structure is estimated as a function of the flight profile. The interceptor is a multi–stage missile that uses fused target location data provided by two ground–based radar sensors and two low earth orbit infrared sensors. Interceptor requirements and limitations are derived as a function of its initial position from the target launch point and the launch delay. Sensor requirements are also examined as a function of the signal–to–noise ratio during the target flight. Electronic attack considerations within the boost phase are also addressed including the use of decoys and noise jamming techniques. The significance of this investigation is that the system components within a complex boost phase intercept scenario can be quantified and requirements for the sensors can be numerically derived.

14. SUBJECT TERMS Boost-Phase Ballistic Missile Intercept, Modeling, Simulation, Missile Requirements, Sensor Requirements, Electronic Attack Effects, Proportional Navigation, Radar Cross Section, IR Energy Radiation Estimation, RF Sensors, IR Sensors, Data Fusion, Decoys, Noise Jamming	**15. NUMBER OF PAGES** 163
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

REQUIREMENTS AND LIMITATIONS OF BOOST–PHASE BALLISTIC MISSILE INTERCEPT SYSTEMS

Kubilay Uzun
Captain, Turkish Air Force
B.S., Turkish Air Force Academy, 1993

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

and

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2004

Author:	Kubilay Uzun
Approved by:	Phillip E. Pace Thesis Advisor
	Murali Tummala Co–Advisor
	John P. Powers Chairman, Department of Electrical and Computer Engineering
	Dan C. Boger Chairman, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this thesis is to investigate the requirements and limitations of boost phase ballistic missile intercept systems that contain an interceptor and its guidance sensors (both radar and infrared). A three–dimensional computer model is developed for a multi–stage target with a boost phase acceleration profile that depends on total mass, propellant mass and the specific impulse in the gravity field. The radar cross–section and infrared radiation of the target structure are estimated as a function of the flight profile. The interceptor is a multi–stage missile that uses fused target location data provided by two ground–based radar sensors and two low earth orbit infrared sensors. Interceptor requirements and limitations are derived as a function of its initial position from the target launch point and the launch delay. Sensor requirements are also examined as a function of the signal–to–noise ratio during the target flight. Electronic attack considerations within the boost phase are also addressed including the use of decoys and noise jamming techniques. The significance of this investigation is that the system components within a complex boost phase intercept scenario can be quantified and requirements for the sensors can be numerically derived.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my wife Ozum for her patience and support.

I would like to thank to my thesis advisors Professor Phillip E. Pace and Professor Murali Tummala for their help to conduct this research. Also I would like to thank to Professor Bret Michael and the ballistic missile defense team for their valuable ideas.

This work was supported by the Missile Defense Agency.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This research investigated the basic requirements and limitations of boost phase ballistic missile intercept systems. In order to accomplish this, a computer code was developed to model a variety of system characteristics including motion in three–dimensional space. After defining the ballistic missile (referred to as 'target' in the text) and the interceptor (referred to as 'missile' in the text) in detail, the radio frequency (RF) and infrared (IR) sensor characteristics and their ability to guide the missile to the target were explored. The normal operation of the boost phase ballistic missile intercept system was tested for several scenarios. Finally, the effects of the possible use of electronic attack on the defense system, which a ballistic missile target may employ during the boost phase, were investigated.

Developing a computer code to simulate the boost phase scenario was the methodology used for this research. Equations were used regarding missile trajectories, propulsion, and sensor calculations to construct a theoretical basis for the research. The computer simulation results for each step were verified by using simple cases. Then, all verified parts of the system were brought together to run the complex cases.

The boost phase intercept scheme was constructed around the following scenario. An intercontinental ballistic missile is launched from a given launch site. The target is tracked by two ground–based RF sensors and two space–based IR sensors. The target position data is transmitted to a fusion processor to calculate an accurate target position. The fused target position data is used to guide a missile. The missile is launched after a certain delay following the target launch and establishes a collision geometry with the target. At a suitable distance, a kill vehicle is launched from the missile to accomplish the intercept. The kill vehicle hits to kill the target, and the intercept is accomplished.

The first step in the development of the simulation was modeling the mechanics of the target. The target was modeled by evaluating the sum of all acting force vectors in the three–dimensional Cartesian coordinate system. The change in mass due to fuel consumption and change in gravity due to the distance from the Earth's center were also considered. Propulsion was modeled by using the consumption rate and the specific impulse

of the fuel used. The trajectory equation and the rocket equation were used as a theoretical basis for the flight of the target. All test runs showed that the computer model reflected the results of the equations satisfactorily. After verification, an example case including an intercontinental ballistic target attack targeting San Francisco, California was conducted, and the measured data yielded valuable findings regarding all physical parameters of the missile during its flight, such as distances, heights, and velocities.

The second step was to model the missile. Proportional navigation in a three–dimensional Cartesian coordinate system was implemented. To verify the results, a simple case with a constant speed target was simulated. Tests showed that the implementation of proportional navigation worked satisfactorily and the target was hit. The missile model was run against the target model developed previously and data were collected and presented. Finally, the zero–lag control system developed so far was extended to a non–zero–lag control system by modeling the missile dynamics with a single time constant, third–order transfer function. The miss distance results due to missile dynamics were presented. Next, the requirements regarding location of the missile were investigated. Test runs yielded good insight in terms of missile capability and location.

The third step was the prediction of target parameters from the sensors' point of view. This included the radar cross section (RCS) for RF sensors and IR radiation estimation for IR sensors. The monostatic RCS was predicted by modeling the target structure with triangular facets. The modeled structure was evaluated by another software program (POFACETS). Calculating the plume IR radiation intensity using Planck's law and integrating the emitted energy in the band of interest summarized the simplistic IR energy radiation estimation.

The fourth step was modeling of the sensors. Optimal location for the RF sensors was determined. The transmission delay between the target track data collection and missile guidance was modeled, and their effect was investigated. A set of RF sensor parameters was proposed, and the effect of different radar design parameters was investigated in detail. The resulting radar specifications were utilized to quantify the signal–to–noise ratio during the intercept. The RMS error in angle and range were quantified using the computer model. The probabilistic nature of target positional error was presented. IR sen-

sor design parameters were discussed. Two IR sensors were located at a low Earth orbit, and a probabilistic target positional error introduced by the IR sensors was quantified. Data fusion was implemented by averaging target track inputs. The fused track data were then used to guide the missile, and results were presented. The effect of tracking quality on miss distance was investigated.

The final step was the investigation of electronic attack effects in the boost phase. A common assumption is that a ballistic missile has enough time and opportunity to attack a defense system electronically by using many measures, such as using multiple warheads, decoys, and/or metallized balloons, or disguising its IR signature by cooling or shrouding the warhead after the boost phase, but the missile does not prioritize the electronic attack while it is still intact and accelerating. However, there is no reason for the ballistic missile not to perform this type of attack, although it is technically more complicated. To explore the electronic attack effects, the decoy trajectory was modeled in the simulation, and separation of the decoy due to acceleration of the missile was shown. The effect of IR and RF decoys was investigated. The amount of chaff dipoles required to screen the target was calculated. The effect of reacquisition following a track transfer to decoy was also examined as well as the effect of noise jamming.

Milestones used to construct the model led to many results and contributions. Mechanical models of the target and the missile unearthed many requirements and limitations along with the ability to choose the capability and location of the defense system elements. The work also shed light on the effectiveness of the common electronic attacks, such as IR and RF decoys as well as noise jamming.

Results reported here were significant since the boost phase intercept scenarios have not been investigated previously as much as the other phases. The computer code uses a three dimensional Earth centered system that other researchers can easily use to implement different scenarios. Deduction of missile parameters in terms of capability and position may contribute to the future decisions on ongoing national missile defense plans. Examination of RF and IR sensor parameters and locations are also significant. Finally, investigating the electronic attack during the boost phase answers many questions while raising more questions for future investigations.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BALLISTIC MISSILE DEFENSE

Defending the United States against a long–range ballistic missile attack has been an issue for many years. The ballistic missile defense plans arose many times and in many forms during the past 50 years. According to Fowler, it appears once every 17 years in different forms [Ref. 1]. The Strategic Defense Initiative (SDI), also known as the Star Wars Project, was proposed during the Reagan administration in 1985. Efforts using laser–based defense programs intensified in the late 1980s and have continued with the airborne laser (ABL) most recently [Ref. 2].

As a result of the observations on the World's new members of the missile club, the United States has enacted the Congressional National Missile Defense Act in 1989 stating that a program to defend the Untied States against the missile attacks is accepted as policy [Ref. 1].

During the last decade, since the collapse of the USSR in 1991, the questions regarding the ability of a small country having the intent and capability to hit the United States with long–range ballistic missiles have been asked increasingly. Donald Rumsfeld presented a report to Congress in 1998 discussing the presence of this kind of threat [Ref. 3]. The report pointed out the growing market in the area fed by uncontrolled know–how and personnel unleashed by the collapse of the USSR and motivated by money [Ref. 3].

Before the report, it was generally believed that individual missile technologies developed by small countries would tend to be original. However, many analysts now believe that the proliferation of missiles is much more likely by using simpler methods, such as brain drain, transferring technology into the country or simply buying them [Ref. 4].

It is very well known that the Rumsfeld report states that Iran or North Korea can achieve such a capability within five years of deciding so. Whether or not this is an overestimation, the report raises crucial questions. Can a small state obtain such a capability in the near future? What is the motivation for such a state for doing so? According to Oberg, passion for third world countries to seize on the capability of launching rockets serves three main objectives: the ability to carry warheads onto the territory of their opponents, contributing to space applications, and becoming a world power [Ref. 4]. According to tables provided by Zakheim, lesser powers having ballistic missiles with a range of more than 1,000 km are India, Iran, Israel, North Korea, Pakistan and Ukraine [Ref. 3].

North Korea is one of the countries attracting special consideration. Does North Korea have such an intention or capability? North Korea's ICBM development program is evident in terms of intent as well as capability. Interrogation of a defector has unearthed that the final objective of North Korea is to build missiles capable of hitting the United States. Taepo–dong II missiles are believed to have been improved for carrying large payloads to 4,000–6,000 km while they may have the capability of carrying lighter payloads up to 10,000 km [Ref. 4]. North Korea's No–dong test launch in 1993 focused the attention of scientists worldwide on the increasing capability of this country. Although it was claimed by North Korea that this was a launch intended to carry a satellite into orbit, as of yet, no one has found any evidence of this satellite. Whether or not it was a failed satellite launch or a trick to hide a near–intercontinental ballistic missile test, the launch showed that the threat is imminent [Ref. 4].

Although the threat is evident and confidence about defending the United States from any conventional, nuclear, chemical, or biological attack is absolute, many scientists disagree with the existing road map of the National Missile Defense (NMD) efforts. A debate continues on the NMD program. Some of the issues, which Fowler has pointed out, are as follows. Firstly, realization of such a capability to neutralize existing intercontinental hit capability of Russia and China may drive them to build more capabilities. Secondly, launching an intercontinental ballistic missile may not be the first priority method to place a conventional, nuclear, chemical, or biological threat in the United States. Finally, it is far from convincing that the existing ballistic missile defense technology is capable of doing the intended job [Ref. 1].

Regarding the threat priority, although the September 11th disaster has changed the threat perception radically, some observers feel that recent announcements indicate that the Bush administration is far from stopping the efforts regarding an intercontinental attack, which may be executed by ballistic missiles [Ref. 5].

Capability is another issue. Some feel that the existing NMD plans are considered incomplete and nothing more than a deterrence tool [Ref. 1]. Why are ongoing plans considered unsatisfactory? Most of the problems associated with the existing plan are related to the mid–course interception, and the mid–course ballistic missile intercept still occupies a majority of resources used by the Ballistic Missile Defense Organization [Ref. 5].

A common approach is to detect the target by using space–based IR sensors, which can sense the huge amount of energy emitted from the rocket plume. After detection, a target track is established by ground–based RF sensors (radars) in coordination with IR sensors to generate useful and accurate target position data to guide an interceptor. Since an intercontinental ballistic missile should travel roughly 10,000 km and fly 30–40 minutes, it makes perfect sense to look for the capability to intercept the ICBM for the midcourse or terminal phase before it hits the target. As detailed below, however, several researchers have shown that this is not as straightforward as it seems.

Although mid–course intercept of ballistic missiles has several advantages, such as the ability to locate assets at home and adequate time for detection, decision and interception, it has more drawbacks. The disadvantages can be summarized as being more susceptible to electronic attack, probability of debris landing in friendly territory if the warhead is not completely destroyed and possibility of the defense system being overwhelmed by utilization of submunitions instead of a single warhead [Ref. 5].

Assuming that the threat will complete its flight intact would be dangerously oversimplifying the problem. Additionally, there are many types of electronic attack, which can be used by the ballistic missile during its flight. Lewis and Postol list these as multiple submunitions, decoys, radar and infrared stealth by shrouding, and maneuvers [Ref. 6].

The MIT Countermeasures Report also emphasizes similar issues that can be summarized as follows. Firstly, there is no reason to believe that a country capable of building and launching a ballistic missile can also exploit an electronic attack. Secondly, an electronic attack might very likely affect, overwhelm or fail the planned NMD system. Finally, an electronic attack may include submunitions, false targets including replica decoys, decoys using signature diversity, and decoys using anti–simulation (metallized balloons, shrouds, chaff, electronic decoys), radar signature reduction, infrared stealth, hiding the warhead, and maneuver [Ref. 7].

Historical lessons learned show that the attacker does not need to possess the sophisticated technology as the defender to defeat the defense system. During the 1991 Gulf War, probably unintentional breakup and tumbling of al–Husayn missiles resulted in the almost total failure of Patriot defenses [Ref. 6]. All solutions associated with the post–boost–phase defense must consider a common fact that when the acceleration of the missile ends, the possibility is great for the deployment of different electronic attacks. For long–range ballistic missiles, each deployed particle from the main payload follows the same trajectory regardless of its mass. Thus, a small chaff dipole weighing on the order of grams is not different from a heavy warhead in outer space where atmospheric effects can be neglected.

This situation naturally directs minds to another option, which is the boost–phase intercept. Boost phase is defined as the initial stage of the ballistic missile flight lasting from missile launch to the burnout of rocket engines [Ref. 5]. During the powered flight where the missile is still intact, technical considerations differ [Ref. 6]. The boost–phase ballistic missile intercept, although not affected by the factors associated with the other phases, has other problems to manage. These are usually detection and decision requirements, which in turn, result in a need to locate the interceptors very close to the target launch site.

Another crucial question arises immediately: What if the electronic attack is executed during the boost phase? The resulting point concerning the NMD efforts is the assumption that by using a boost phase intercept plan, most of the problems associated with the mid–course intercept could be resolved. How valid is this assumption? What are the strong and weak points of the defense system against this kind of attack?

B. PRINCIPAL CONTRIBUTIONS

This research investigated the basic requirements and limitations of boost phase ballistic missile intercept systems, which was accomplished by developing a computer code to model a variety of system characteristics including motion in three–dimensional space. After defining the ballistic missile (referred to 'target' in the text) and the interceptor (referred to as 'missile' in the text) in detail, the radio frequency (RF) and infrared (IR) sensor characteristics and their ability to guide the missile to the target were explored. The normal operation of the boost phase ballistic missile intercept system was tested for several scenarios. Finally, the effects of the possible use of electronic attack on the defense system, which a ballistic missile target may employ during the boost phase, were investigated.

The methodology used for this research consisted of developing a computer code to simulate the boost phase scenario. The complexity of the interacting dynamics in the scenario usually limits the opportunity to define all elements with simple equations. Therefore, a step–by–step approach was followed to verify the accuracy of the results. Equations were used regarding missile trajectories, propulsion, and sensor calculations to construct a theoretical basis for the research. The computer simulation results for each step were verified by using simple cases. Then, all verified parts of the system were combined to run the complex cases.

The boost phase intercept scheme was constructed around the following scenario. An intercontinental ballistic missile is launched from a given launch site. The target is tracked by two ground–based RF sensors and two space–based IR sensors. The target position data is transmitted to a fusion processor to calculate an accurate target position. The fused target position data is used to guide an intercept missile. The missile is launched after a certain delay following the target launch and establishes a collision geometry with the target. At a suitable distance, a kill vehicle is launched from the missile to accomplish the intercept. The kill vehicle hits to kill the target, and the intercept is accomplished.

Two important elements of the scenario are beyond the scope of this research. The first is the data fusion. The track data are fused here by using a simple averaging method. The second is kill vehicle flight. The missile is allowed to fly until it hits the target instead of launching a kill vehicle.

The work reported here consists of many results and contributions. A three–dimensional computer model was developed for a multi–stage target with a boost phase acceleration profile that depends on total mass, propellant mass and the specific impulse in the gravity field. The radar cross–section and infrared radiation of the target structure was estimated as a function of the flight profile. The interceptor is a multistage missile using fused target location data provided by two ground–based radar sensors and two low–earth–orbit (LEO) infrared sensors. Interceptor requirements and limitations were derived as a function of its initial position from the target launch point and the launch delay. Sensor requirements were examined as function of the signal–to–noise ratio (SNR) during the target flight. Electronic attack considerations within the boost phase are also addressed, including the use of decoys and noise jamming techniques. Mechanical models of the target and the missile unearthed many requirements and limitations with the ability to choose the capability and location of the defense system elements. The work also shed light on the effectiveness of the common electronic attacks, such as IR and RF decoys as well as noise jamming.

The significance of this investigation is that the system components within a complex boost phase intercept scenario can be quantified and requirements for the sensors numerically derived. The computer code uses a three–dimensional Earth–centered system that other researchers can easily use to implement different scenarios. Deduction of missile parameters in terms of capability and position may contribute to future decisions for ongoing national missile defense plans. Examination of RF and IR sensor parameters and locations are also significant. Finally, investigating the electronic attack during the boost phase answers many questions while raising more questions for future investigation.

C. THESIS OUTLINE

Chapter II is dedicated to the development of the code and modeling the mechanics of the target. The scenario where an intercontinental ballistic missile attacks San Francisco, California is conducted, and the results presented in terms of the target parameters, such as distances, heights, and velocities.

Chapter III models the missile. The proportional navigation is implemented in three–dimensional Cartesian coordinate system. The missile model is run against the target model developed previously, and data are collected and presented. After finishing modeling missile mechanics, the requirements regarding locating the missile are investigated. Results of test runs are presented.

Chapter IV predicts target parameters from the sensors' point of view. The monostatic RCS is predicted by modeling the target structure with triangular facets. The modeled structure was evaluated by another software program (POFACETS). Calculating the plume radiation intensity by using Planck's law and integrating the emitted energy in the band of interest summarize the simplistic IR energy radiation estimation.

Chapter V models the sensors. Optimal locations for the RF sensors are found. This chapter models the transmission delay between the target track data collection and missile guidance. A set of RF sensor parameters is proposed, and the effects of different radar design parameters on tracking quality are investigated in detail. The model locates two IR sensors in a LEO and quantifies the probabilistic target positional error introduced by the sensors. The fused track data are used to guide the missile, and the results are presented.

Chapter VI investigates the effect of electronic attack in the boost phase. To explore the electronic attack effects, the simulation models the decoy trajectory, and also the separation of the decoy due to acceleration of the missile. The effect of IR and RF decoys is investigated. The amount of chaff dipoles required to screen the target is calculated. The effect of reacquisition following a track transfer to decoy is also examined as well as the effect of noise jamming.

Chapter VII provides the concluding remarks.

Appendix A shows a detailed chart for the code flow. Appendix B provides a complete listing of the MATLAB code developed for this research.

II. TARGET MODELING

This chapter presents a three–dimensional target model that operates in the Earth's gravity field. The simulation models a multi–stage, boosting target capable of reaching the velocity of 6.5 km/s that enables it to reach intercontinental distances. The basic definitions and assumptions for the model, the coordinate systems used, the gravity field effects, and the target velocity requirements are discussed as follows.

A. BASIC DEFINITIONS AND ASSUMPTIONS

The target body obeys Newton's Second Law that can be defined in vector form as

$$\vec{F} = m\vec{a} \tag{2-1}$$

where $\vec{F}$ is the force vector (in N) acting on the center of gravity (CG) of the target body, m is the total mass (in kg), and $\vec{a}$ is the net acceleration vector (in $\mathrm{m/s^2}$). There are only two types of major force vectors acting on the CG of the target body. These are the thrust $\vec{T}$ and the weight $\vec{W}$. The net force vector $\vec{F}_{net}$ can be written as

$$\vec{F}_{net} = \vec{T} + \vec{W}. \tag{2-2}$$

The thrust vector is assumed to be in the direction of the velocity vector $\vec{v}$. In the model, the direction of the thrust vector is not modified (i.e., thrust is not vectored) meaning that the target makes a gravity turn [Ref. 8: p. 255]. To develop the thrust vector magnitude T, the stage specific impulse I_{sp} (in s) is first expressed as [Ref. 8:p. 255].

$$I_{sp} = \frac{-T}{\dot{W}} \tag{2-3}$$

where change in weight over time $\dot{W}$ can be defined as a function of change in mass over time or in–stage fuel consumption dm/dt (in kg/s) and gravitational acceleration at the current distance from the center of the Earth g (in m/s^2) as

$$\dot{W} = \frac{dm}{dt} g. \tag{2-4}$$

Substituting (2-4) into (2-3) and solving for T gives

$$T = -\frac{dm}{dt} g I_{sp}. \tag{2-5}$$

The stage specific impulse is assumed to be constant, and fuel is assumed to decrease linearly during the stage.

The weight vector is in the direction of the center of the Earth. The magnitude of the weight vector W can be written as

$$W = mg. \tag{2-6}$$

Since most of the interception occurs in the exoatmospheric region, drag is neglected. Also, since the scope of this study is only on the boost phase, which occurs at relatively small distances from the Earth and short times with respect to the overall target flight, the Earth is assumed to be a perfect non–rotating sphere with a radius of 6,370 km. The above definitions and assumptions were used to build the model.

B. COORDINATE SYSTEMS

Three coordinate systems are used within the model.

The first is the Earth–centered, Earth–fixed (ECEF) Cartesian coordinate system. In the ECEF Cartesian coordinate system, all computations occur in three orthogonal axes. The Earth is located at the origin. In a right–handed system, the positive x–axis passes through 0° N, 0° E, the positive y–axis passes through 0° N, 90° E, and the positive z–axis passes through 90° N. All elements defined in different coordinate systems are translated into the ECEF Cartesian coordinate system. In the following pages, only the name Cartesian coordinate system refers to the ECEF coordinates. Figure 2–1 illustrates the Cartesian coordinate system and the Earth's location.

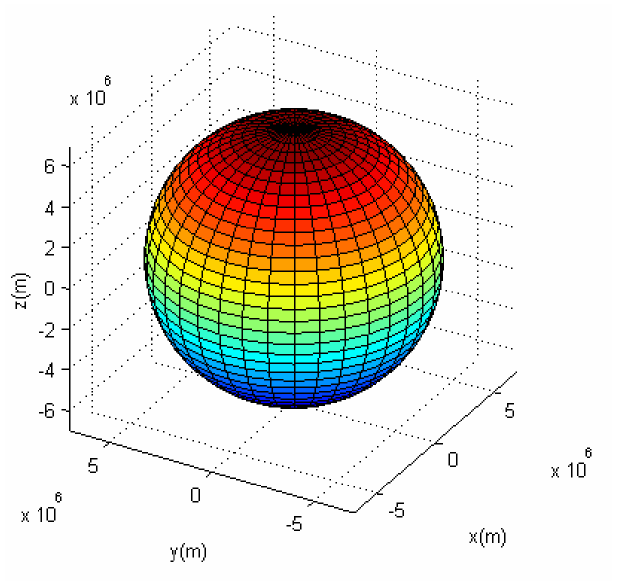


Figure 2–1. The Basic Reference for the Simulation, Cartesian Coordinate System and the Earth's Location.

The second is the geodetic coordinate system. All locations including the target, the missile, and the sensors are defined in the geodetic coordinate system in N/S dd°mm.mmm E/W dd°mm.mmm format. The target launch site contained in the model is located at N 41°00.000 E 129°00.000 and represents the Kilju–kun missile base, North Korea.

The third is the topocentric–horizon coordinate system [Ref. 9:p. 53]. Launch angles are defined in the topocentric–horizon coordinate system where the first element, azimuth, is measured from true north in degrees, and the second element, elevation, is measured from the local horizon in degrees. The topocentric–horizon notation is useful in defining the initial direction of target velocity vector and is independent of the target location. As with all other vectors, target launch angles are also translated to the Cartesian coordinate system before computations.

C. THE GRAVITY FIELD

When the concern is intercontinental ranges, the flat Earth approximation with a constant gravitational acceleration is no longer valid. The direction of the weight vector is towards the Earth's center (round Earth model), and the change in the gravitational acceleration (in $\mathrm{m/s^2}$) is modeled as [Ref. 10:p. 326]

$$g = \frac{GM}{r^2} \tag{2-7}$$

where G is the gravitational constant [Ref. 10: p. 323], which has the approximate value of 6.67×10^{-11} $\text{m}^3/(\text{kg} \cdot \text{s}^2)$, M is the Earth's mass [Ref. 10:p. A–4], which has the approximate value of 5.98×10^{24} kg, and r is the distance from the center of the Earth (in m) assuming that the Earth is a uniform–density, non–rotating, perfect sphere.

Given the launch angle γ and the initial distance r_0 from the center of the Earth, the target distance r (in m) as a function of the central angle θ can be calculated by using the trajectory equation as [Ref. 8:p. 235]

$$r = \frac{r_0 \lambda \cos^2 \gamma}{1 - \cos\theta + \lambda \cos\gamma \cos(\theta + \gamma)} \tag{2-8}$$

where the parameter λ depends on the initial range r_0, launch velocity V, gravitational constant G, and the Earth's mass M as given by [Ref. 8:p. 234]

$$\lambda = \frac{r_0 V^2}{GM}. \tag{2-9}$$

The central angle θ is defined as the angle between the initial launch position and the position of the target in flight measured at the center of the Earth as shown in Fig. 2–2.

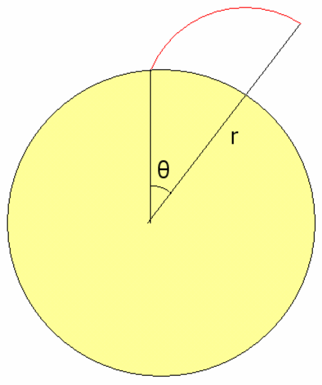


Figure 2–2. Definitions for the Trajectory Equation, the Central Angle θ and the Range r.

Given the theoretical values of distance versus height by the trajectory equation, it is possible to test the gravity field behavior of the model. Note that, with a specified launch velocity and launch angle, the target trajectory is independent of the mass. When setting the thrust of the model to 0, the initial velocity to V and the launch elevation angle to γ, the theoretical and simulated trajectories must match. For an initial velocity of $V = 0.91$ km/s (3,000 feet/s) and a launch angle of $\gamma = 45°$, theoretical and simulation results are illustrated in Fig. 2–3, which shows that they match exactly.

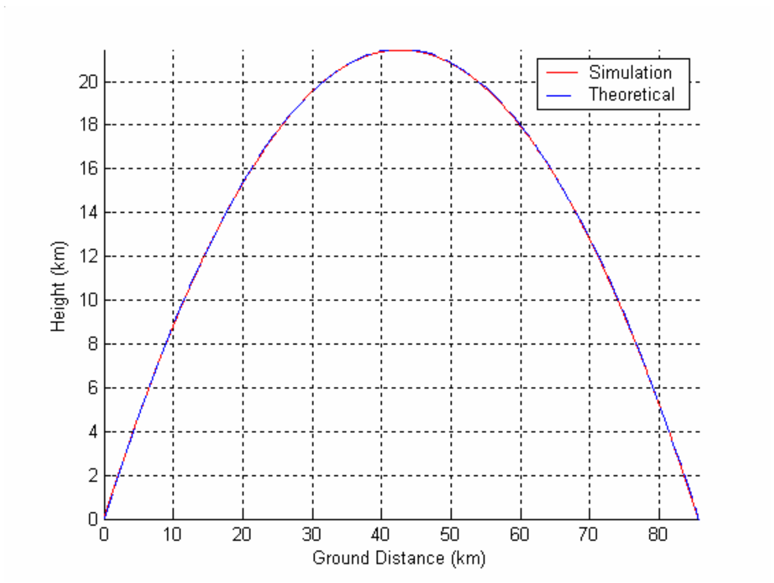


Figure 2–3. Target Trajectory with a Launch Speed of 0.91 km/s (3,000 feet/s) and Launch Angle of 45°.

Figures 2–4 and 2–5 show the target trajectories with initial velocities of $V = 1.83$ km/s (6,000 feet/s) and $V = 7.32$ km/s (24,000 feet/s), respectively, while keeping the launch angle $\gamma = 45°$. In summary, Figures 2–3, 2–4, and 2–5 show that the target trajectory in the model's gravity field yields accurate results indicating that the simulation curves are exactly the same as the curves plotted using (2-8). Velocities of 3,000, 6,000 and 24,000 feet/s are chosen to compare gravity field modeling with the findings given in [Ref. 8:pp. 225–233].

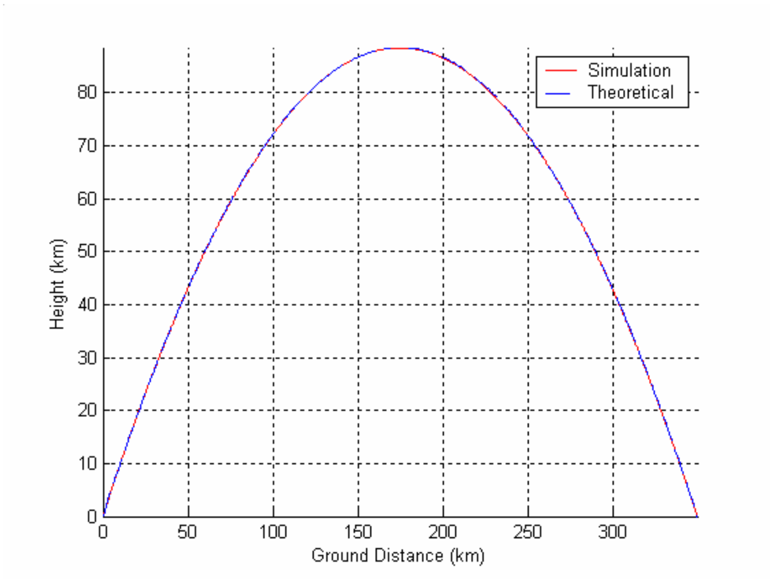


Figure 2–4. Target Trajectory with a Launch Speed of 1.83 km/s (6,000 feet/s) and Launch Angle of 45°.

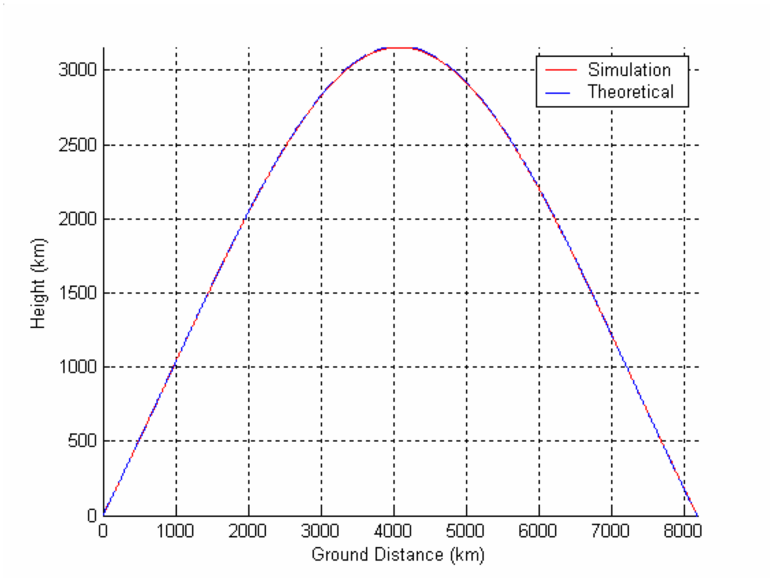


Figure 2–5. Target Trajectory with a Launch Speed of 7.32 km/s (24,000 feet/s) and Launch Angle of 45°.

D. TARGET VELOCITY REQUIREMENTS

The required velocity (in $\mathrm{m/s}$) to hit a target at a specified distance along the Earth's surface is given by [Ref. 8:p. 242]

$$V = \sqrt{\frac{GM(1-\cos\phi)}{r_0 \cos\gamma[(r_0 \cos\gamma / r_e) - \cos(\phi+\gamma)]}} \tag{2-10}$$

where r_e is the radius of the Earth (in m) and the total central angle traveled ϕ (in rad) can be calculated as [Ref. 8:p. 241]

$$\phi = \frac{d}{r_e} \tag{2-11}$$

where d is the specified distance (in m) along the Earth's surface to accomplish the hit.

It is assumed that the ICBM is to be launched at an initial velocity of V from sea level. Given the distance of the ICBM's target d, it is possible to calculate the initial velocity of the ICBM using (2-10). Figure 2–6 illustrates the velocity requirements for various target distances. As the required distance to be hit increases, the required velocity of the target increases. The velocity requirements for two major cities chosen from the East and West Coasts of the United States are illustrated. An ICBM launched from Kilju missile base, North Korea has to travel 8,668 km at true heading 050° to hit San Francisco, California, and 10,771 km at true heading 020° to hit Washington, D.C. To reach this range, the ICBM should be launched at a velocity of 6.95 km/s, and 7.32 km/s for San Francisco, California and Washington, D.C., respectively.

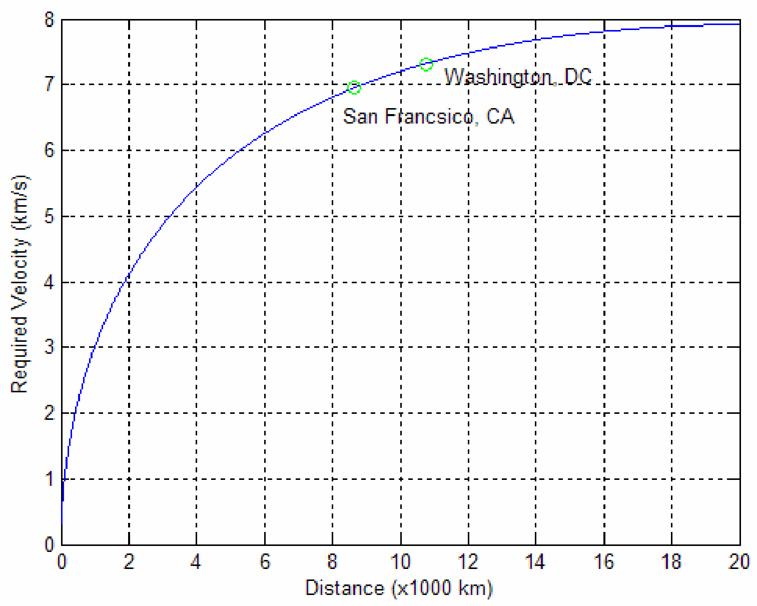


Figure 2–6. Target Velocity Requirements to Hit a Given Distance.

Note that, when modeling a boosting target, velocity requirements will be slightly different since the target has already traveled some ground distance and altitude at burn-out. However, the initial velocity launch model provides good insight into the velocity requirements of the target to be modeled.

E. BOOSTING TARGET MODELING

When the concern is the mid–course or re–entry phase of an intercontinental ballistic missile (ICBM), the initial velocity/launch angle model may provide an adequate basis for simulations. However, for boost phase intercept models, this approach is no longer useful. ICBMs that can threaten the United States burn out in about 3–4 minutes reaching a velocity of 6–7 km/s. Speeds required for hitting targets at certain distances can be computed by using (2-10). However, ICBM design is beyond the scope of this research. The aim is to achieve a realistic boost–phase trajectory and speed profile for a target capable of hitting the cities discussed above.

In the simulation, the boosting capability is modeled by a minimal set of parameters including total and propellant masses as well as the specific impulses and stage burn times. Generic target models are used; however, all parameters are extracted from actual missile specifications using the U.S. Peacekeeper missile [Ref. 11]. Table 2–1 illustrates how the target is modeled. This table is called the data matrix, and there is one of these for each target and missile in the simulation. The target with the data matrix shown in Table 2–1 is capable of boosting up to 6.5 km/s at burnout and, if launched from Kilju Missile Base, North Korea, will hit the West Coast of the United States (specifically, San Francisco). This is a three–stage target and the total mass and dimension of each stage is the same as the U.S. Peacekeeper missile [Ref. 11], with 85% of the total mass of each stage being the propellant mass. Each stage is assumed to be using a fuel with a specific impulse of 300 s and burnout time of 60 s. The total boost phase takes three minutes. This target is assumed to be carrying a payload of 5,000 lbs.

	Stage–1	**Stage–2**	**Stage–3**	**Payload**
Total Mass (lb)	108,000	61,000	17,000	5,000
Propellant Mass (lb)	91,800	51,850	14,450	0
Specific Impulse (s)	300	300	300	0
In–stage Burn Time (s)	60	60	60	0

Table 2–1. Target Data Matrix.

1. Silo Exit Velocity

When launched, the target travels inside the silo (the length of the silo is the length of the missile). Assuming that the target travels vertically with a constant acceleration during this phase, the target speed at silo exit v (in m/s) can be written as

$$v = \int_0^{\tau_e} a\,dt = a\tau_e \tag{2-12}$$

where a is the constant target acceleration, and τ_e is the time when the target exits its silo (in s). In this case, the distance traveled can be written as

$$l = \int_0^{\tau_e} v\,dt = \int_0^{\tau_e} a\tau_e\,dt = a\tau_e^{\;2} \tag{2-13}$$

where l (in m) is the target length. The acceleration a (in $\mathrm{m/s^2}$) can also be written as

$$a = \frac{F_{net}}{m_0} = \frac{T_0 - W_0}{m_0} = \frac{T_0 - m_0 g_0}{m_0} \tag{2-14}$$

where F_{net} is the net force acting on the target, m_0 is the target mass (sum of the total mass of each stage), T_0 is the thrust, W_0 is the weight, and g_0 is the gravitational acceleration at launch. By substituting (2-14) into (2-13), the silo exit time τ_e (in s) can be found as

$$\tau_e = \sqrt{\frac{l m_0}{T_0 - m_0 g_0}} \tag{2-15}$$

By substituting (2-15) into (2-12), the silo exit velocity v can be found as

$$v = \frac{T_0 - m_0 g_0}{m_0}\sqrt{\frac{l m_0}{T_0 - m_0 g_0}} \tag{2-16}$$

The simulation model starts with the silo exit velocity as the initial velocity of the target. For the target definition, this value is approximately 18 m/s.

2. The Rocket Equation and Consequences

The increase in velocity ΔV (in m/s) provided by a single–stage rocket is given by the rocket equation [Ref. 8:p. 247] as

$$\Delta V = I_{sp} g \ln\left(\frac{1}{1-m_f}\right) \tag{2-17}$$

where the mass fraction m_f is defined as [Ref. 8:p. 248]

$$m_f = \frac{W_p}{W_p + W_s} \tag{2-18}$$

where W_p (in N) is the propellant weight, and W_s (in N) is the structural weight including the non–propellant part of the target and payload.

The rocket equation conveys that the achieved velocity by a certain rocket can be enhanced by increasing the specific impulse (or exhaust velocity) and/or increasing the mass fraction. The mass fraction can be increased by reducing the structural parts of a rocket other than the propellant and/or reducing the payload.

The most important consequence of the rocket equation is the fact that a larger missile does not necessarily mean a faster missile. In order to make a missile faster, weight efficiency should be increased or a lesser amount of payload should be used. In this target model, 85% of mass of any stage is assumed to be the propellant. In reality, the actual mass fraction is a larger percentage than that used in this model.

To increase weight efficiency, staging is used [Ref. 8:p. 249]. It has been demonstrated that, as the number of stages approaches infinity, the total weight required to obtain the desired velocity is minimized [Ref. 8:p. 251]. It has also been concluded that use of three stages yields results very close to the optimal [Ref. 8:p. 251]. Since no reason exists to believe that the evolution of potential targets will be less than optimal, a three–stage model is used.

In a three–stage rocket, the mass fraction of separate stages can be calculated by using

$$m_{f,n} = \frac{m_{p,n}}{\sum_{i=n}^{3} m_{t,i} + m_{pay}} \tag{2-19}$$

where n is the stage number, $m_{p,n}$ (in kg) is the stage propellant mass, $m_{t,i}$ (in kg) is the stage total mass, and m_{pay} (in kg) is the payload mass. In a given stage, all weights except the propellant weight of that stage is the structural weight. Each of the three mass fractions yields a separate ΔV where the total velocity capability of the overall system is given by

$$\Delta V = \sum_{i=1}^{3} \Delta V_i \tag{2-20}$$

where ΔV_i is obtained by substituting (2-19) into (2-17) and the overall increase in the velocity is obtained by summing individual increases for each stage.

To prove that the boosting target simulation works satisfactorily, the theoretical speeds computed using the rocket equation are compared to those of the simulation. To accomplish this, gravity field effects are removed from the system temporarily. Table 2–2 lists the theoretical values from (2-17). Computations show that the target reaches a theoretical speed of 1.925 km/s at the end of Stage–1, 4.811 km/s at the end of Stage–2, and 7.96 km/s at burnout. These computations assume that no gravity field is present and a constant exhaust velocity of 2943 m/s (which is a product of the sea level gravitational acceleration and the specific impulse) is used.

	Stage–1	**Stage–2**	**Stage–3**
Stage Mass Fraction	0.480	0.625	0.657
Stage ΔV (km/s)	1.925	2.886	3.149

Table 2–2. Theoretical Velocity Capability of Target Model.

The simulation model computes the acceleration by evaluating (2-5). It continuously integrates the acceleration to compute the velocity and continuously integrates velocity to compute the position of the target. Test runs of the simulation yielded speeds of 1.928 km/s at the end of Stage–1, 4.811 at the end of Stage–2, and 7.959 km/s at the end of Stage–3. Small errors in ΔV (maximum of 3 m/s) between the theoretical and the simulation values are due to the small initial velocity of the target (other than zero) and the integration error. Comparison of theoretical and simulation speeds shows that the acceleration capability of this target model reflects the theory satisfactorily.

F. BOOSTING TARGET IN THE GRAVITY FIELD

So far, the gravity field performance of the model where the target has zero thrust and the boosting performance of the model in the lack of gravity field have been validated. Thus, two major forces were investigated acting on the body independently. In the light of tests and findings, it is possible to conclude that the boosting target in the gravity field works satisfactorily.

In practice, theoretical speeds cannot be reached since the work is done against gravity. By running the simulation under real conditions, major aspects of the target trajectory were examined. Figures 2–7 through 2–11 are results of a simulation of an ICBM attack from Kilju Missile Base, North Korea against San Francisco, California with an initial launch angle of $\gamma = 84^{\circ}$. Figure 2–7 is a three–dimensional illustration of the attack on Earth's surface.

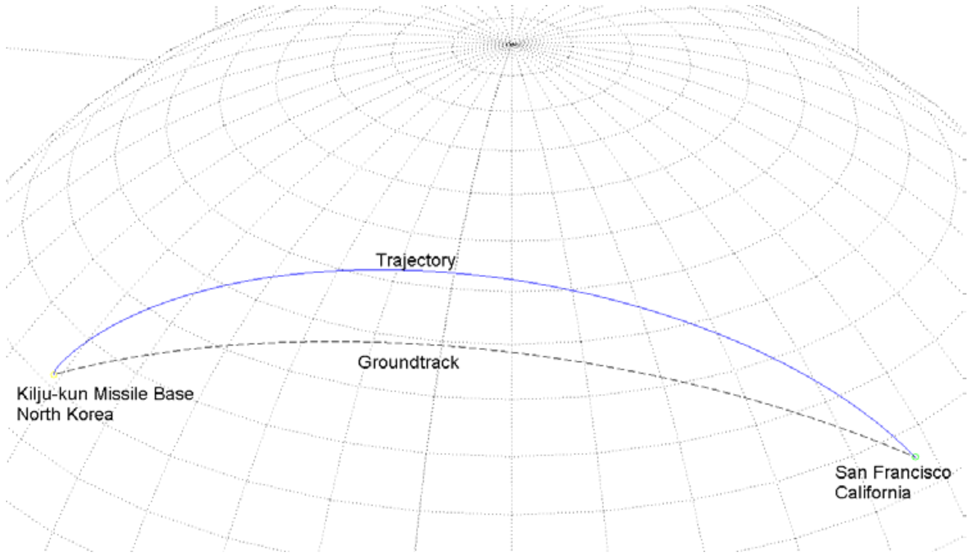


Figure 2–7. 3D Overview of an ICBM Attack from Kilju-kun Missile Base, North Korea to San Francisco, California.

Figure 2–8 shows the traveled height versus ground distance. The target hits the ground at approximately 8,640 km, reaching an apogee of approximately 1,560 km. This plot shows the realistic performance of the target by including the gravity field and realistic thrust parameters.

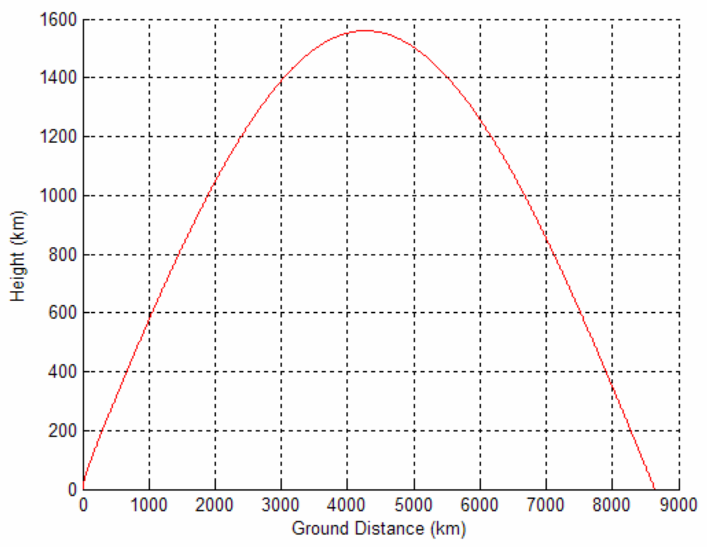


Figure 2–8. Ground Distance versus Height for the San Francisco Attack.

Figure 2–9 shows the velocity profile for the entire flight. The target has reached a velocity of approximately 6.5 km/s at the end of the boost phase. After burnout, it decelerates due to gravity until the apogee is reached. Followed by that, the target begins to accelerate due to gravity.

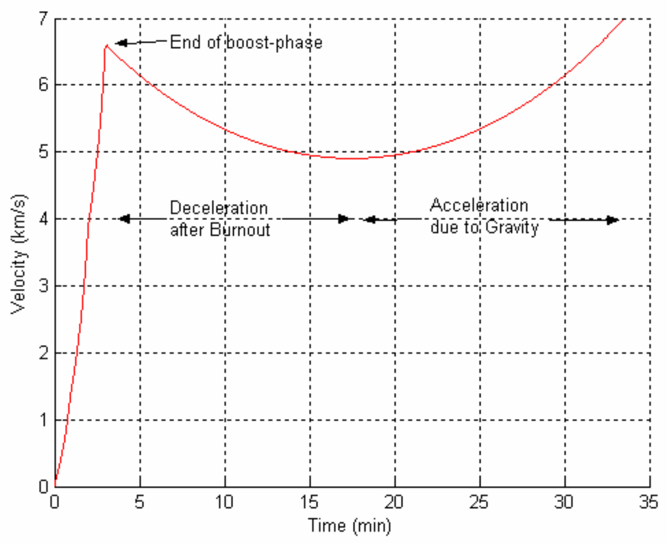


Figure 2–9. Velocity versus Flight Time for the San Francisco Attack.

A closer look at the boost phase part of the velocity versus flight time plot in Fig. 2–9 reveals the acceleration profile due to staging as shown in Fig. 2–10. The simulation results for this specific run yielded 1.43 km/s speed and 26 km altitude at the end of Stage–1, 3.86 km/s speed and 107.5 km altitude at the end of Stage–2 and 6.53 km/s speed and 250.5 km altitude at burnout. The velocities reached in simulation are lower than the theoretical (non–gravity) velocities. The target has a unique acceleration profile coming from its individual stage thrusts and fuel consumption. Since fuel consumption and thrust are constant during a stage, the in–stage acceleration increases as the fuel is consumed and the weight is decreased.

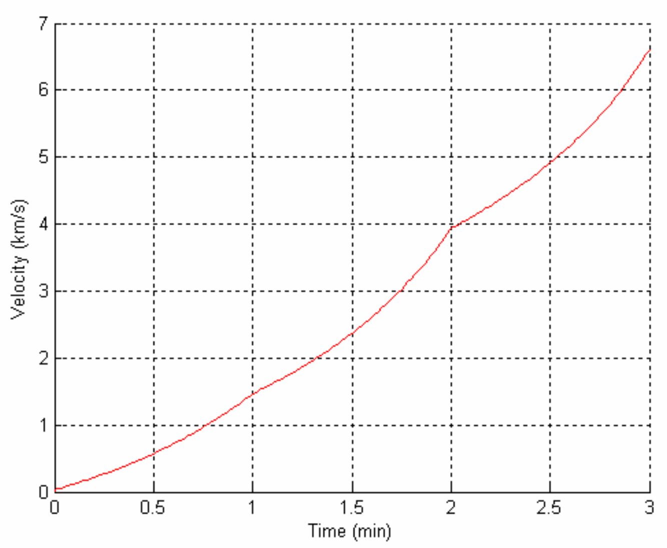


Figure 2–10. Velocity versus Flight Time for the San Francisco Attack (Boost Phase Only).

Figure 2–11 illustrates the change in mass during the boost phase. The mass decreases linearly during each stage due to constant fuel consumption. However, stage transitions have discontinuities. Discontinuities are a result of canister jettisoning at the end of the stage. After the boost phase, the target continues with the payload only.

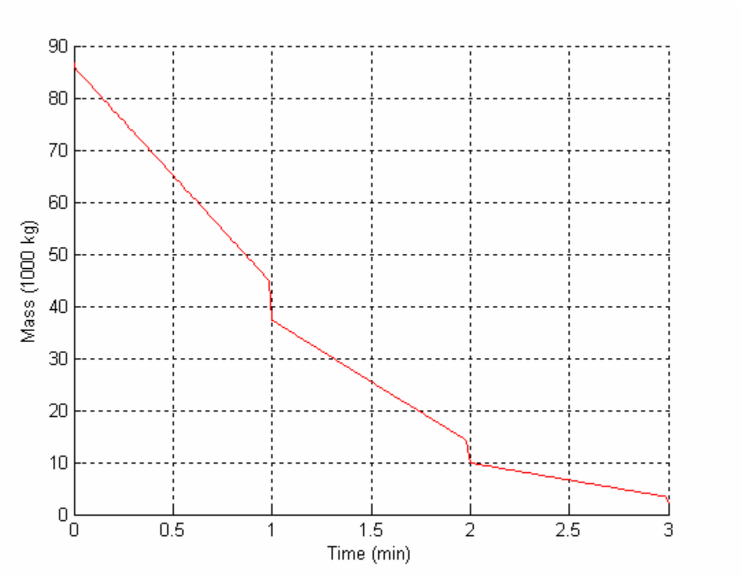


Figure 2–11. Total Mass versus Flight Time for the San Francisco Attack (Boost Phase Only).

G. SUMMARY

This chapter developed a three–dimensional boosting target model. Equations regarding gravity field and thrust were used to construct a theoretical basis for this model. Later, the simulation was run many times for different cases in order to compare the simulation results with the theoretical values. All tests showed that, under given assumptions, the 3D target model works satisfactorily. This is an important step for developing the boost phase intercept simulation. The simulation was run under realistic conditions in an example of an intercontinental attack, and data were collected. The resulting graphs provided an understanding of the boost and the other phases of the attack. The next step is to examine the missile characteristics.

THIS PAGE INTENTIONALLY LEFT BLANK

III. INTERCEPTOR MISSILE MODELING

In this chapter, a three–dimensional multi–stage interceptor missile model that operates in Earth's gravity field is developed. The boosting interceptor is capable of intercepting a multi–stage boosting target within the boost phase with a minimum lateral acceleration and small miss distance. Below, the basic definitions and assumptions are given along with a description of the missile guidance and dynamics.

A. BASIC DEFINITIONS AND ASSUMPTIONS

The basic rules used to develop the target model also apply to the interceptor missile model. The missile operates under two major force vectors, the thrust $\vec{T}'$ (in N), and the weight $\vec{W}'$ (in N). The weight vector $\vec{W}'$ always acts in the direction towards the center of the Earth similar to the target model. The thrust vector $\vec{T}'$ is also aligned with the velocity vector with the exception that its direction is modified to obtain the guiding force (lateral acceleration). In this case, the net force vector $\vec{F}'_{net}$ acting on the body can be written as

$$\vec{F}'_{net} = \vec{T}'_v + \vec{T}'_p + \vec{W}' \tag{3-1}$$

where $\vec{T}'_v$ (in N) is the thrust component along the direction of velocity vector $\vec{v}$ (in m/s) and $\vec{T}'_p$ (in N) is the thrust component perpendicular to the velocity vector $\vec{v}$. The overall magnitude of the two thrust components parallel and perpendicular to the velocity vector is always equal to the total thrust $\vec{T}'$ provided by the rocket engine. Other definitions and assumptions used for the target model also apply to the missile model.

B. BOOSTING MISSILE MODELING

For the realistic target model developed in Chapter II, a target design capable of obtaining approximately 6.5 km/s at burnout was presented. It was shown that this target design can hit the West Coast of the United States. A missile capable of intercepting this target has been designed. Velocities required for this type of intercept are greater than the ballistic target velocity as detailed below.

The missile design starts with the same set of parameters as were used to define the target boosting capability; however, the design is more efficient and has a smaller payload. Table 3–1 summarizes the missile parameters used here. This is a three–stage missile having total masses and dimensions the same as the U.S. Peacekeeper missile [Ref. 11]; however, 95% of the mass of each stage is assumed to be the propellant mass. Each stage uses a fuel with a specific impulse of 300 s and burn time of 60 s. The total boost phase takes three minutes. This missile carries a payload of 1500 lbs. Note that the payload of the interceptor missile is the kill vehicle. The objective of the missile is to carry the kill vehicle to an optimal position in space to allow it to complete the intercept.

	Stage–1	**Stage–2**	**Stage–3**	**Payload**
Total Mass (lb)	108,000	61,000	17,000	1,500
Propellant Mass (lb)	102,600	57,950	16,150	0
Specific Impulse (s)	300	300	300	0
In–stage Burn Time (s)	60	60	60	0

Table 3–1. Missile Data Matrix.

The same principles used in the design of the target are used. To achieve a more efficient performance, mass fractions are improved and the payload is reduced. The same launch elevation angle of $\gamma = 84°$ is used for better comparison with the target performance.

Test runs of the simulation yielded 1.845 km/s at the end of Stage–1, 5.17 km/s at the end of Stage–2, and 10.31 km/s at the end of stage–3. These velocities are obtained without the guidance force applied and reflect the free–flight performance of the missile. When guided, the energy used to guide the missile trajectory effectively reduces the obtainable velocities.

C. MISSILE GUIDANCE

Proportional navigation is used for missile guidance. The proportional navigation is optimal for constant velocity targets [Ref. 12]. It is emphasized here that the classical proportional navigation guidance law is suboptimal for the boost phase intercept type of application. Against accelerating targets, it has been shown that saturation is always reached near interception [Ref. 13]; however, the terminal phase of the intercept is out-side the scope of this research. The missile's objective is to carry the kill vehicle to a suitable position to terminate the intercept. Although the terminal phase (kill vehicle) was not investigated, the missile flies until it passes the target in order to measure the miss distance and assess the effectiveness of the guidance algorithm.

Figure 3–1 shows a block diagram of missile processing. The missile takes the position vector of the target $\vec{r}_t$ and computes the line of sight (LOS) vector $\vec{\lambda}$ by subtracting its own position vector $\vec{r}_m$. The LOS vector $\vec{\lambda}$ is differentiated to calculate the LOS rate vector $\dot{\vec{\lambda}}$ and closing velocity V_c. The calculated parameters $\dot{\vec{\lambda}}$ and V_c are multiplied by the navigation coefficient N' to calculate the commanded lateral acceleration vector $\vec{n}_c$. The flight control system uses the commanded lateral acceleration to change the attitude of the missile resulting in the achieved lateral acceleration vector $\vec{n}_L$. The achieved lateral acceleration vector $\vec{n}_L$ is integrated along with the other accelerations acting on the system resulting in a new missile position $\vec{r}_m$.

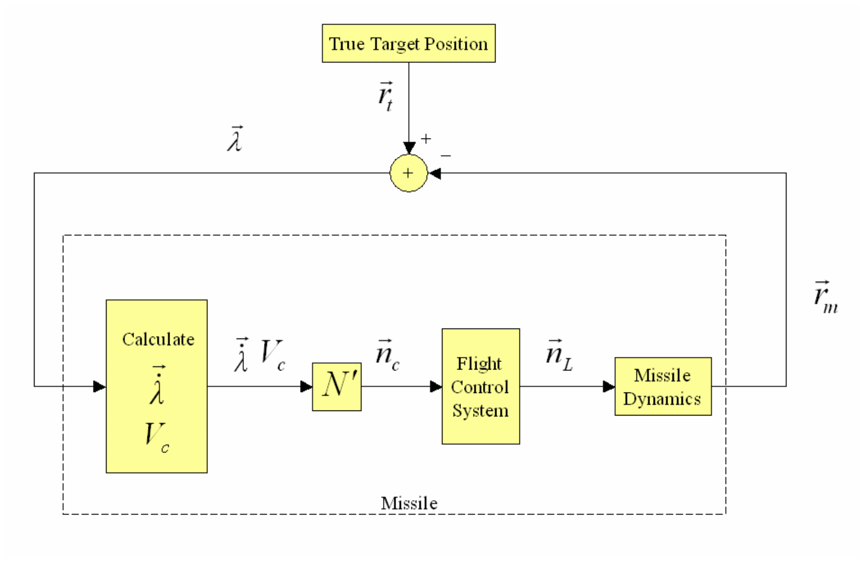


Figure 3–1. Missile Block Diagram.

For proportional navigation, the commanded acceleration is applied perpendicular to the LOS and given in scalar form as [Ref. 8:p. 12]

$$n_c = N'V_c\dot{\lambda}. \tag{3-2}$$

Proportional navigation relies on the LOS being constant or the LOS rate being zero. In other words, the missile and the target are on a collision triangle. The seekers used in tactical missiles usually provide the LOS rate. The missile design investigated computes its own guidance commands using the position data supplied by off–board sensors via a data link.

The instantaneous LOS vector is computed first as

$$\vec{\lambda} = \vec{r}_t - \vec{r}_m \tag{3-3}$$

The instantaneous LOS vector is normalized to obtain the LOS unit vector $\hat{\lambda}$. In the next sample time, the new LOS is computed by using (3-3) and also converted to the unit vector. Vector subtraction of these two unit vectors is the direction in which the acceleration command is applied and is always perpendicular to the instantaneous LOS. After normalizing the acceleration command, the unit vector is

$$\hat{n}_c = \frac{\hat{\lambda} - \hat{\lambda}_{previous}}{\left|\hat{\lambda} - \hat{\lambda}_{previous}\right|} = \frac{\Delta\hat{\lambda}}{\left|\Delta\hat{\lambda}\right|} \tag{3-4}$$

where $\hat{n}_c$ is the unit acceleration command vector perpendicular to the LOS, $\hat{\lambda}$ is instantaneous unit LOS vector, and $\hat{\lambda}_{previous}$ is the previous unit LOS vector. Note that this is only the direction of the acceleration command to be applied for guidance. The magnitude of the LOS rate can be obtained by

$$\left|\dot{\lambda}\right| = \frac{\left|\Delta\hat{\lambda}\right|}{\Delta t} \tag{3-5}$$

where Δt is the simulation step time.

The closing velocity V_c is also required and computed as a range rate. The range between the missile and target is the magnitude of the LOS vector $\left|\vec{\lambda}\right|$. This magnitude is calculated for each step time of the simulation and differentiated. Dividing the difference in the range by the simulation step time yields closing velocity as

$$V_c = \frac{\left|\Delta\vec{\lambda}\right|}{\Delta t}. \tag{3-6}$$

The magnitude of the commanded acceleration is computed by multiplying the navigation ratio (unitless constant), the closing velocity (scalar), and magnitude of the LOS rate. Multiplying the magnitude of the acceleration command with the acceleration command unit vector yields the commanded acceleration command vector. This can be written as

$$\vec{n}_c = \hat{n}_c N' V_c \left|\dot{\lambda}\right| \tag{3-7}$$

For a zero lag system, the achieved acceleration n_L is always equal to the commanded acceleration n_c and, for the moment, it is assumed that the missile dynamics are free of lags.

The computed acceleration command is perpendicular to the LOS; however, missile acceleration commands can only be applied perpendicular to the missile attitude or the velocity vector. Thus, only the commanded acceleration component perpendicular to the velocity vector contributes to the missile guidance.

To ignore the parallel component and calculate the perpendicular component, the following procedure is used. First, the angle β between the commanded acceleration vector and the velocity vector is calculated as

$$\beta = \cos^{-1}(\hat{n}_c \bullet \hat{v}) \tag{3-8}$$

where $\widehat{v}$ is the unit velocity vector. Next, the acceleration vector component parallel to the velocity vector is obtained as

$$\left|\vec{n}_{c\|}\right| = \left|\vec{n}_c\right| \cos\beta \tag{3-9}$$

The acceleration vector parallel to the velocity vector can be calculated by multiplying the velocity unit vector and the magnitude of the commanded acceleration vector component parallel to the velocity vector as

$$\vec{n}_{c\|} = \hat{v}\left|n_{c\|}\right| \tag{3-10}$$

The acceleration vector component perpendicular to the velocity vector is obtained by subtracting the parallel component from the original acceleration vector as

$$\vec{n}_{c\perp} = \vec{n}_c - \vec{n}_{c\|}. \tag{3-11}$$

The commanded acceleration vector $n_{c\perp}$ can be applied by vectoring the thrust (movement of the nozzle), control surfaces, or lateral thrusters at the CG of the missile. The required thrust component perpendicular to the velocity vector is then

$$\vec{T}'_p = \vec{n}_{c\perp} m_m \tag{3-12}$$

where m_m is the missile mass at the current sample time.

From (3-1), the magnitude of the thrust component along the velocity vector is

$$\left|\vec{T}'_v\right| = \sqrt{\left|\vec{T}'\right|^2 - \left|\vec{T}'_p\right|^2}\,. \tag{3-13}$$

1. Guidance System Against Constant Speed Target

To ensure that the proportional navigation guidance system is working properly, constant speed missile and target test scenarios are used with the gravity field and thrust deactivated. The target velocity was set to 6.5 km/s and the missile velocity to 10 km/s. The target and the missile were launched in a geometry that introduces a heading error in order to examine the acceleration commands generated. Representative target and missile flights during the test run are shown in Fig. 3–2.

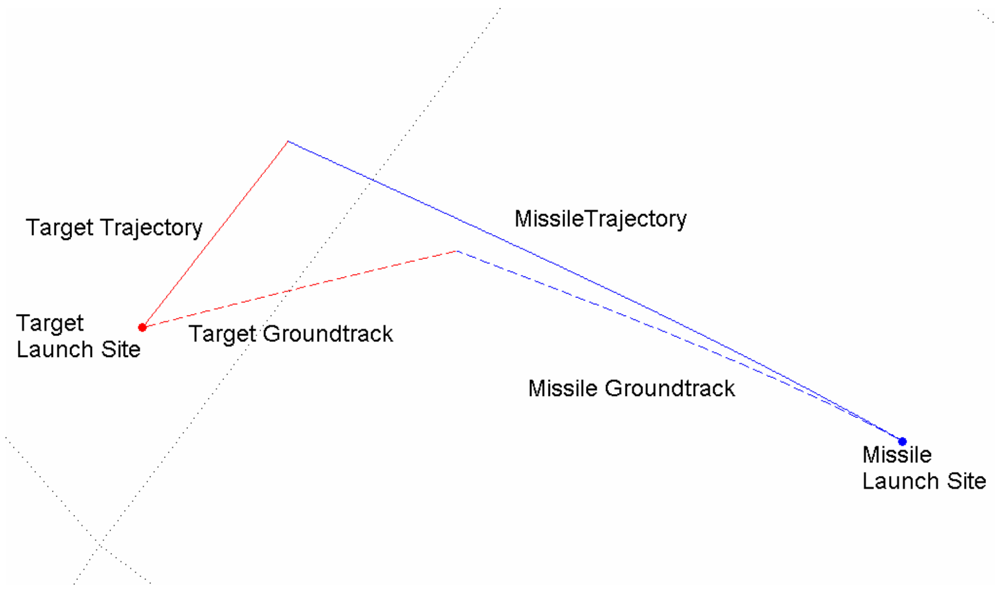


Figure 3–2. 3D Overview of a Typical Intercept for the Constant Speed Scenario.

Figure 3–3(a) shows the height of the interceptor missile and the target as a function of the ground distance. The target and the missile reach an approximate altitude of

145 km at the time of the intercept. The missile travels an approximate ground distance of 420 km while the target travels 250 km since the missile is faster. Figure 3–3(b) shows the velocity of the target and missile as a function of the flight time. Figure 3–3(b) reveals that the missile speed does not change, illustrating that the velocity vector and acceleration commands are orthogonal.

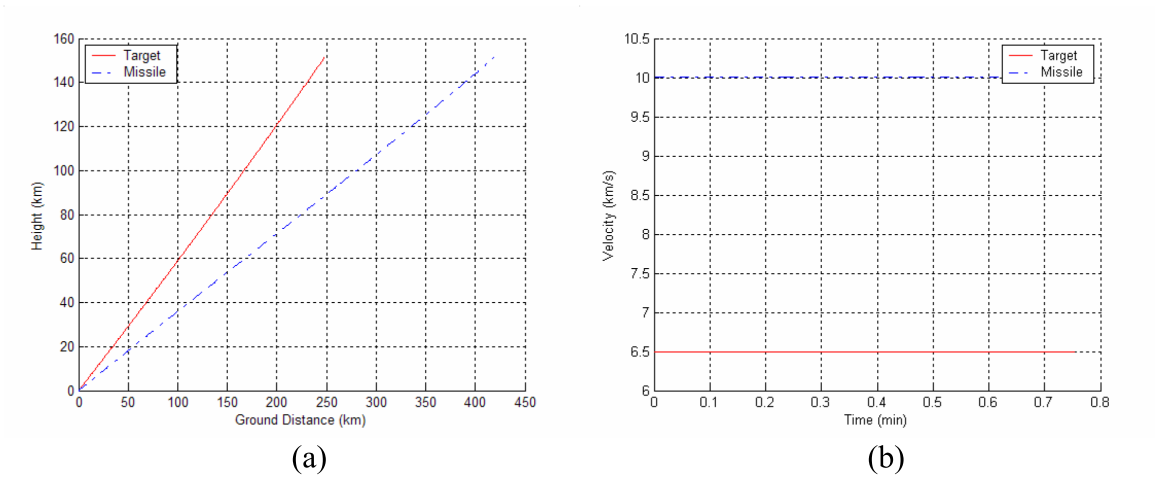
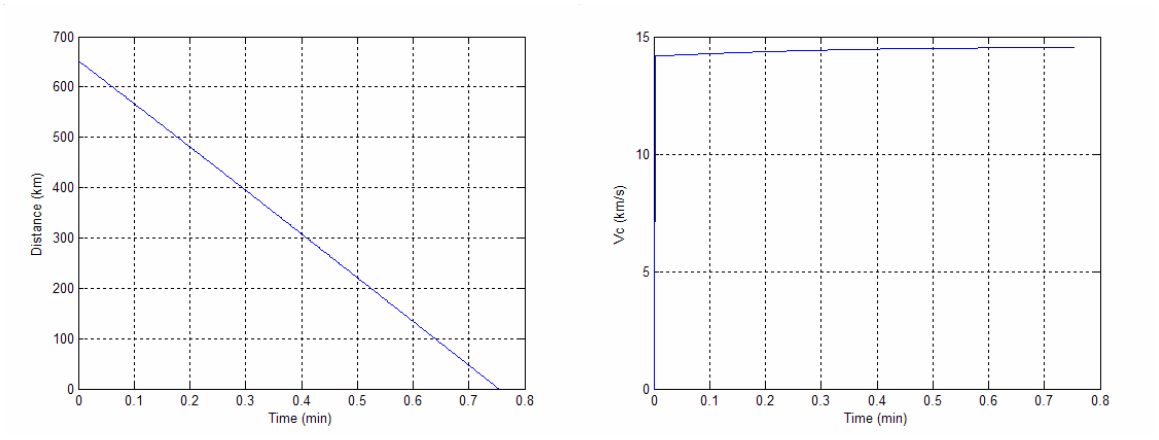


Figure 3–3 The Target and the Missile Flight Characteristics for the Constant Speed Scenario: (a) Ground Distance versus Height, (b) Velocity versus Flight Time.

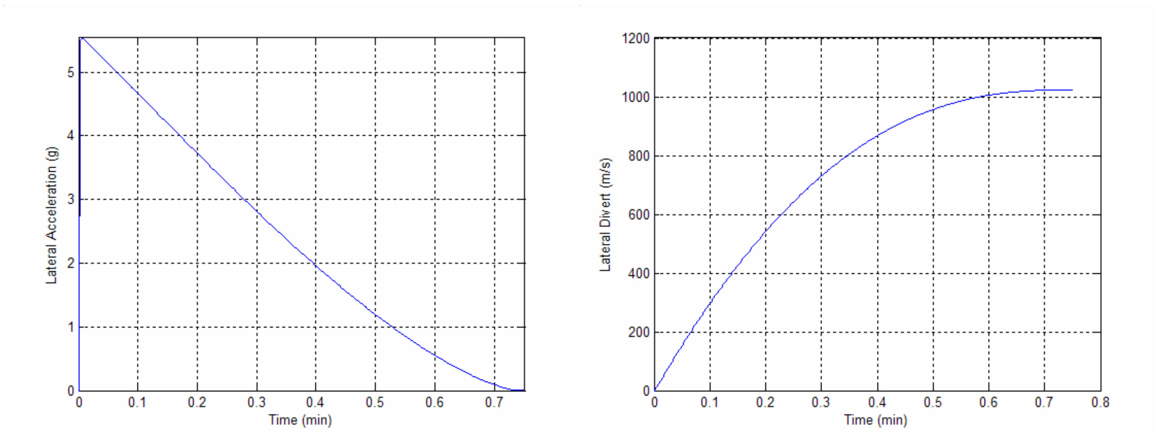
Figure 3–4(a) shows the LOS magnitude between the missile and the target. Figure 3–4(b) shows the closing velocity as a function of time. From Fig. 3–4(a), the range between the missile and the target decreases linearly since they are constant speed bodies. Figure 3–4(b) confirms that as the collision course is established, closure velocity stabilizes as well as the LOS.



(a) (b)

Figure 3–4. The Target and the Missile Closure Characteristics for the Constant Speed Scenario: (a) Range versus Flight Time, (b) Closing Velocity versus Flight Time.

Figures 3–5(a) and Fig. 3–5(b) show the missile lateral acceleration and the missile lateral divert results, which are typical [Ref. 8:pp. 19-23]. The heading error introduced at the beginning of the simulation causes the initial acceleration command and corresponding lateral divert. As the collision course is established, the missile acceleration decreases to zero and the missile hits the target. The collected data after the simulation finished indicates that the target and missile traveled ground ranges of 248 km and 419 km, respectively. The intercept time was 0.755 minutes. The miss distance was under 1 meter, and the final lateral divert was 1023 m/s.



(a) (b)

Figure 3–5. Missile Guidance Characteristics for the Constant Speed Scenario: (a) Missile Lateral Acceleration, (b) Missile Lateral Divert.

In summary, the constant speed target tests showed that the proportional navigation implementation in this missile design works satisfactorily.

2. Guidance System Against ICBM Model

With gravity and thrust activated, the missile model developed here and the target model developed in Chapter II are simulated together to illustrate an interception. The major difference in this type of intercept is the large accelerations provided by both the missile and the target and fluctuations in acceleration due to staging.

Figure 3–6 illustrates a 3D overview of the intercept for the accelerating target. As seen in Fig. 3–6, the trajectories of both the missile and the target are no longer straight lines when compared to Fig. 3–2.

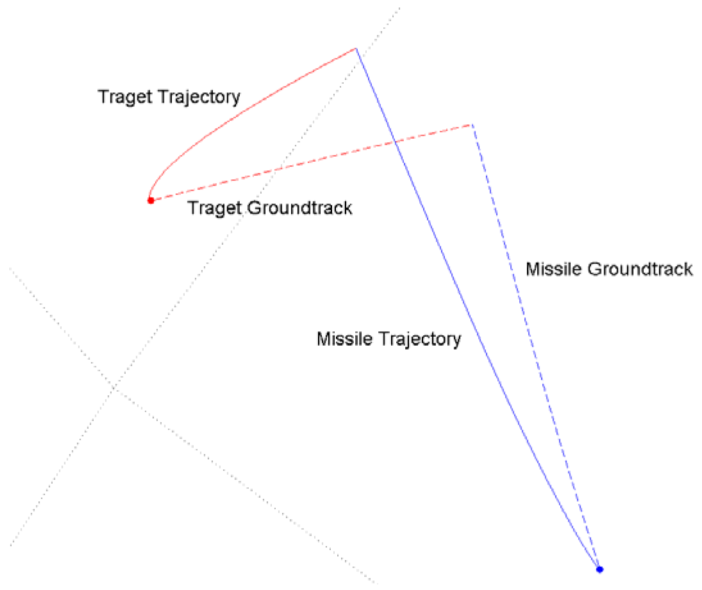


Figure 3–6. 3D Overview of the Intercept for the Accelerating Target.

Figure 3–7 shows the acceleration profile of the target; only acceleration perpendicular to the LOS is relevant and plotted. Target acceleration perpendicular to LOS is also known as target maneuver. Although the target is not maneuvering deliberately, acceleration due to rocket engines and intercept geometry causes the missile to encounter a target maneuver up to 5 g. A bigger problem is the target maneuver discontinuities during stage changes. All these factors cause unexpected guidance commands as shown in the following sections.

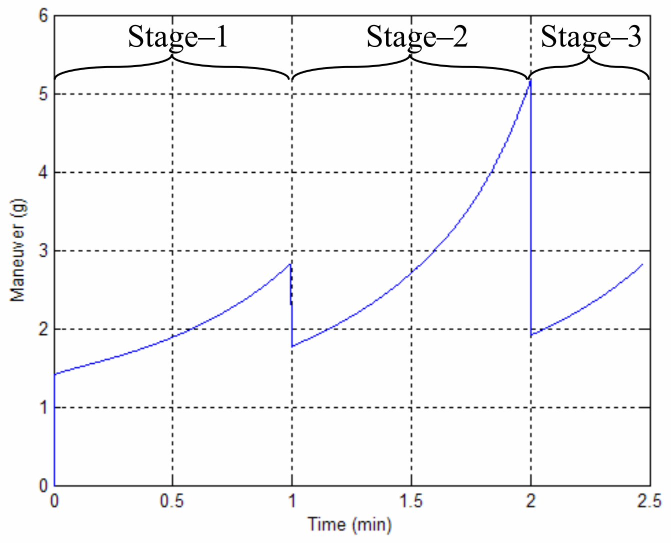
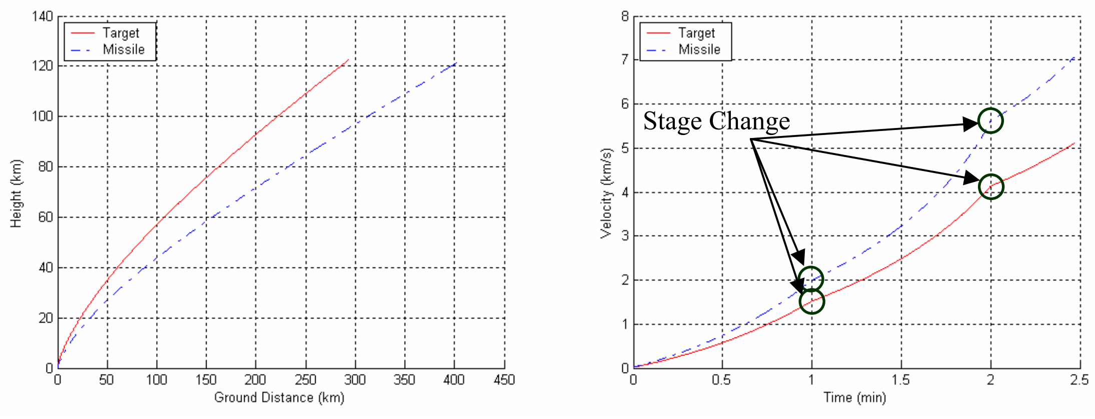


Figure 3–7. Target Maneuver during the Intercept.

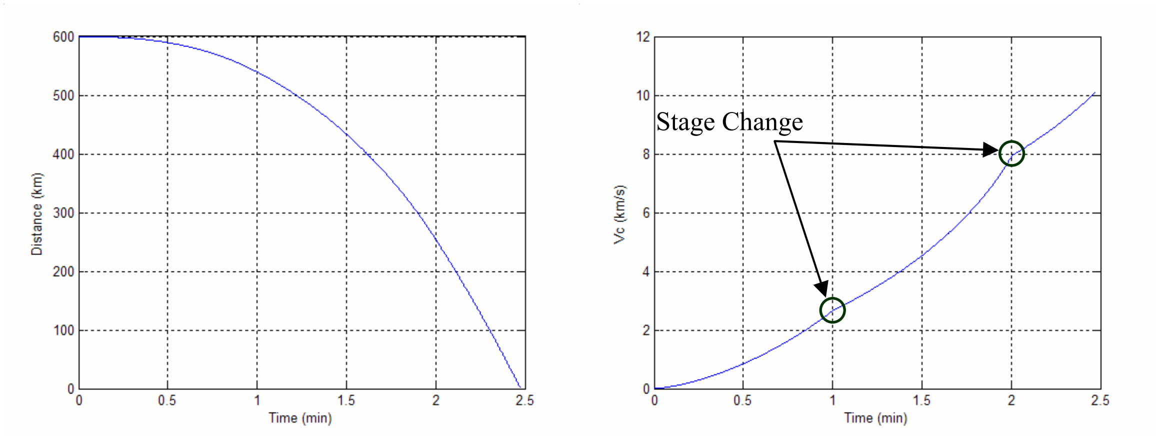
Figure 3–8 shows plots of ground distance versus height and flight time versus velocity for the missile and the target. The target and the missile reach an approximate altitude of 120 km at the time of the intercept. The missile and the target travel an approximate ground distance of 400 km and 300 km, respectively. In Fig. 3–8(b), since the missile is superior to the target in capability, it reaches higher velocities. Also, Fig. 3–8(b) illustrates the velocity profile due to staging. Since the missile and the target are launched synchronously, velocity discontinuities occur at the same time.



(a) (b)

Figure 3–8 The Target and the Missile Flight Characteristics for the Accelerating Target: (a) Ground Distance versus Height, (b) Velocity versus Flight Time.

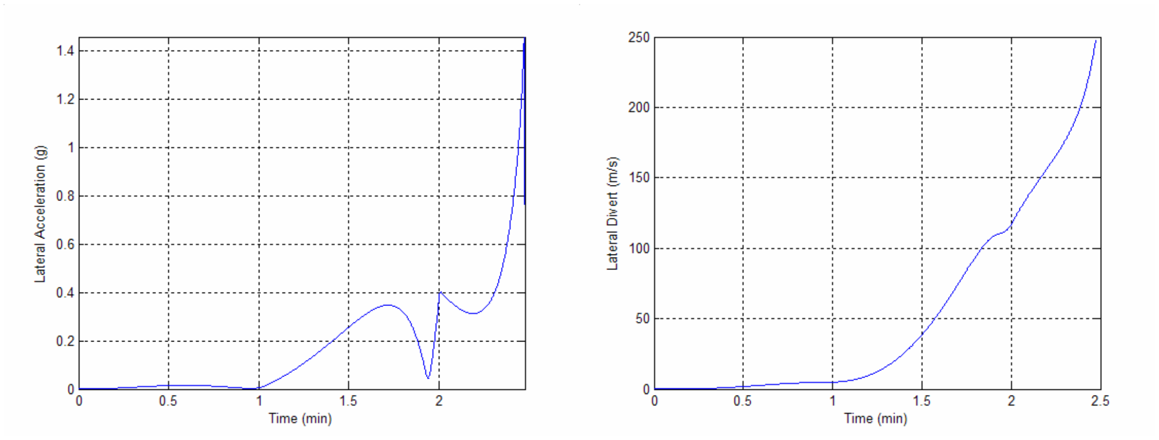
Figure 3–9 illustrates distance and closure velocity during the intercept. As shown in Fig. 3–9(a), the change in the distance is no longer linear since both bodies are accelerating. Fig. 3–9(b) shows the unique closing velocity profile due to acceleration and position of the missile and the target. Note that the closing velocity is approximately 10 km/s at the time of hit. This situation cannot be seen in conventional intercept cases and is a challenging aspect of the boost–phase ballistic missile intercept problem.



(a) (b)

Figure 3–9. The Target and the Missile Closure Characteristics for the Accelerating Target: (a) Missile–Target Distance versus Flight Time, (b) Missile–Target Closure Velocity versus Flight Time.

Figure 3–10 illustrates the lateral acceleration and lateral divert versus flight time. Figure 3–10(a) shows that missile lateral acceleration commands are usually under 0.4 g and increase up to 1.4 g at the terminal phase. Figure 3–10(b) shows that the lateral divert of the missile increases up to 250 m/s. Both results are highly dependent on the initial heading error between the missile and the target at the time of launch. It can be concluded that both results are reasonable and can be achieved by the missile flight control system.



(a) (b)

Figure 3–10. Missile Guidance Characteristics for the Accelerating Target: (a) Missile Lateral Acceleration, (b) Missile Lateral Divert.

In summary, in this simulation, the target and missile traveled a ground distance of 294 km and 407 km, respectively. Intercept time was 2.4766 minutes. Miss distance and total lateral divert were 1.3 m and 248.3 m/s, respectively.

D. FLIGHT CONTROL SYSTEM

So far, the missile guidance system is perfect. In other words, the achieved acceleration n_L is always equal to the commanded acceleration n_c. This type of model is known as a zero–lag guidance system. Since the control system can respond to acceleration inputs immediately, even though the acceleration levels or lateral diverts differ, the miss distance will always be equal to zero [Ref. 8:p. 32].

Guidance systems have lags (or delays) in their response. In this section, the model is expanded to support a practical flight system control response. The system response is modeled as an n^{th} order transfer function (Laplace form). Although it is very common practice to model missile dynamics as a 3^{rd} order transfer function [Ref. 8: p. 98], the model described here is able to support any order. It should be emphasized that the mechanical modeling of the control system dynamics are beyond the scope of this research. The objective was to model the system response given the n^{th} order transfer function time constants. For this reason, arbitrary time constants are used, which can easily be replaced by realistic ones to model a specific missile.

If the system lag is modeled as a 1st order transfer function, the relation between commanded and achieved acceleration can be written as [Ref. 8:p. 32]

$$\frac{n_L}{n_c} = \frac{1}{1+sT} \tag{3-14}$$

where s is the complex frequency and T is the system time constant.

The general form of an nth order all–pole transfer function is written as

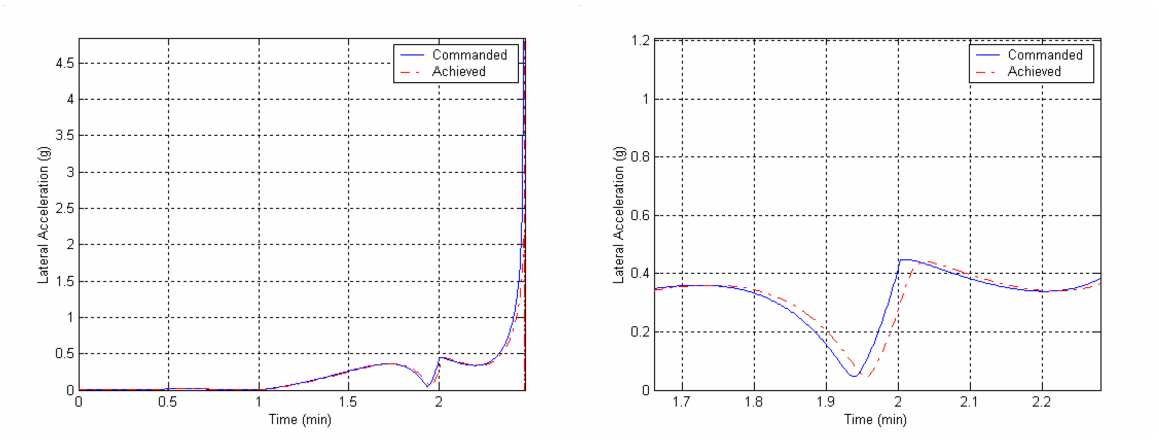
$$\frac{n_L}{n_c} = \frac{a}{bs^n + cs^{n-1} + \ldots + ds^2 + es + f} \tag{3-15}$$

where a , b , c , …, f are constants characterizing the system poles.

The 3rd order single time constant flight control system used within the model has the transfer function

$$\frac{n_L}{n_c} = \frac{1}{\frac{T^3}{27}s^3 + \frac{T^2}{3}s^2 + Ts + 1}. \tag{3-16}$$

Figure 3–11 shows how the system lag affects the commanded versus achieved acceleration for a time constant of $T = 1$s. Figure 3–11(a) shows n_c and n_L for the complete flight, and Fig. 3–11(b) shows a close–up view of the discontinuities. Note that the system response lags behind the control input because the missile model is no longer a zero–lag model. This implies that even if accurate target position data is provided, the missile will experience some miss distance.



(a) (b)

Figure 3–11. (a) Control System Lag, (b) Detail.

To evaluate the miss distance, the 3rd order single time constant system was tested against a constant–speed target. To exclude other effects, the gravity and the thrust were deactivated in the simulation. The missile and target are launched simultaneously with velocities of 10 km/s and 6.5 km/s, respectively. In this scenario, the flight time t_f is approximately 45 seconds. Time constant T is varied from 0 to 45 seconds in 0.1–second increments. Miss distance data was collected for each run. Since the direction of the miss is not of concern, the miss distance measurements are the magnitude of the distance vector at the time of miss. Test runs resulted in the curve shown in Fig. 3–12, which is normalized with respect to the maximum values of each axes.

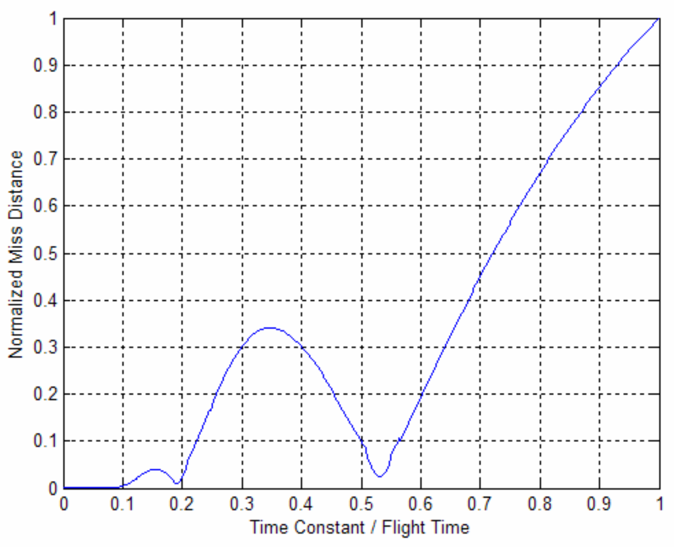


Figure 3–12. Miss Distance versus Time Constant for the Constant Speed Scenario.

From Fig. 3–12, if the time constant is less than one tenth of the total flight time, the miss distance is negligible against a constant–speed target. As the time constant increases, the miss distance reaches certain peaks while continuing to increase. This result conforms to the results reported in the literature [Ref. 8:pp. 31-50] with some differences in the notation. This concludes the efforts in the development of a realistic target–missile model, which will be the basis for the remaining discussion in this thesis.

E. MISSILE REQUIREMENTS

To examine the missile requirements, three different missile models that have different capabilities are defined. The first one has a velocity capability similar to that of the target; the second and the third are superior missiles. Table 3–2 summarizes the parameters defining these missile models.

	Stage Propellant Mass Fraction (% of Total Mass)	**Payload (lb)**	**ΔV at Burnout (km/s)**
Generic Missile–1	85%	5,000	~6.5
Generic Missile–2	90%	3,250	~8
Generic Missile–3	95%	1,500	~10

Table 3–2. Summary of Generic Missile Specifications.

Given the missile capabilities as in Table 3–2, the effect of the missile location on the flight time was investigated. Given the objective that the target should be intercepted in the first three minutes, it is possible to use the simulation to determine the maximum distance between the missile and the target launch site. The best–case scenario happens when the missile is located in the attack direction of the target, and the launch delay equals zero. Usually, this situation cannot be fulfilled due to territorial limitations, and detection/decision requirements; however, examination of flight time under these circumstances shows the theoretical limitations of possible missile site locations.

The simulation was used to examine several scenarios where missiles with different capabilities were located in different distances from the target in the attack direction. For each case, the resulting intercept time was recorded as illustrated in Fig. 3–13. Since the interceptions exceeding the 3–minute limit (total boost phase) are considered failures, the corresponding missile–target site distance where each curve crosses the 3 minute intercept time line can be interpreted as the limitation to the missile launch site location. Figure 3–13 demonstrates that, even in the attack direction, GM–1, GM–2 and GM–3 can never be located more than 941, 1038 and 1140 km from the target launch site, respectively.

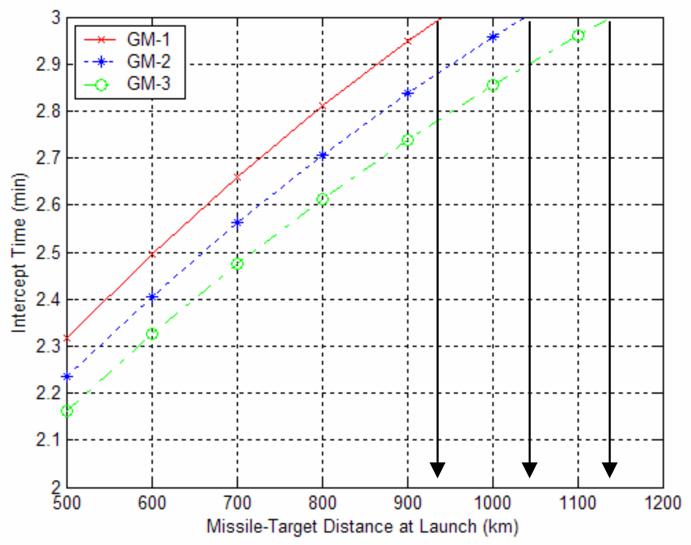


Figure 3–13. Limitation to the Missile Launch Site Distance from the Target Launch Site: Missile Directly at Attack Direction, No Launch Delay.

The missile may not be located directly at the attack direction. As noted in the target modeling, an attack targeting San Francisco, California or Washington, D.C. from the chosen target launch site should be at an approximate true heading of 50° and 20°, respectively. For the scenario where the target is launched from North Korea, these bearings remain inside the territory of Russian Federation. This scenario forces the location of the missile at easterly bearings in the Sea of Japan. Figure 3–14 illustrates the missile location and the probable attack directions. The figure shows that any intercept attempt may very likely encounter angular errors of 40° to 70°. The following investigation assumes the worst–case scenario. If the attack is in the direction of Washington, D.C. and the missile is located east of the target launch site, the missile location is severely constrained because of the angular deviation introduced.

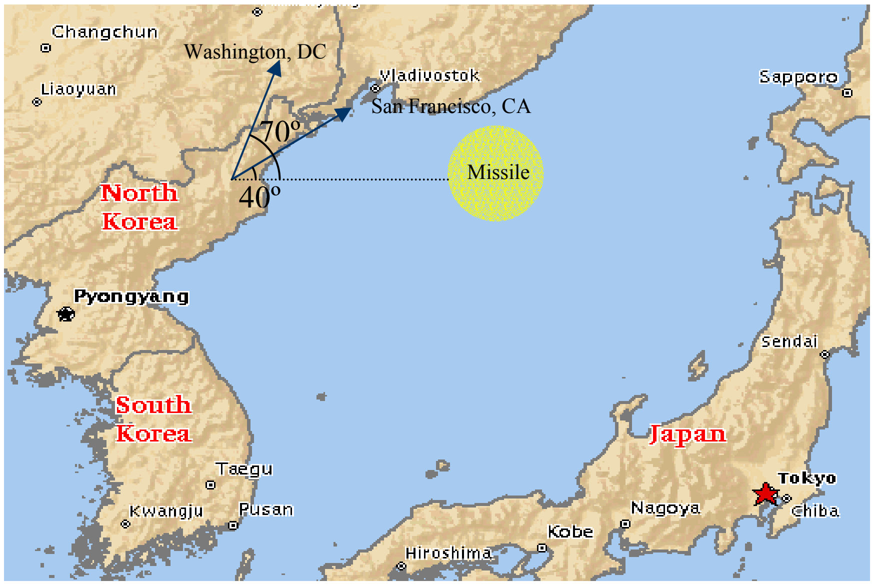


Figure 3–14. Potential Attack Directions and the Missile Location.

Figure 3–15 illustrates the impact of a 70° angular error between the missile position and the attack direction. By using the curves corresponding to different missiles, it can be concluded that GM–1, GM–2 and GM–3 can never be located more than 325, 477 and 593 km from the target launch site, respectively. Figure 3–15 also highlights the fact that the more superior the missile, the more flexibility possible when positioning the missile. By comparing Fig. 3–15 with Fig. 3–13, the introduced angular error approximately halved the required distance for GM–3 while it caused approximately one–third degradation for GM–1. This shows that the superior missile can tolerate location and angular deviations better.

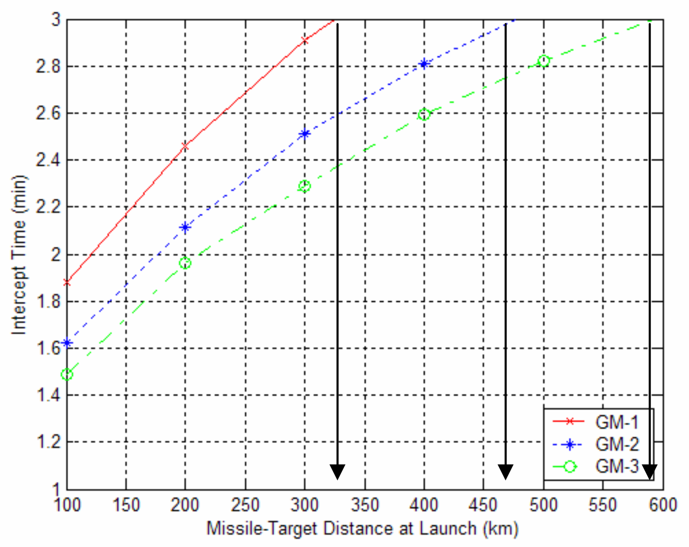


Figure 3–15. Limitation to the Missile Launch Site Distance from the Target Launch Site: 70° Angular Error, No Launch Delay.

Another important factor is the launch delay. Following the target launch, the detection and the decision process to intercept the missile takes place. This missile launch delay also introduces additional limitations. Figure 3–16 illustrates the effect of launch delay for GM–1 for three different missile to target distances when the missile is located exactly in the attack direction. As the distance from the target increases, tolerance to launch delay decreases. For example, if GM-1 is located at a distance of 700 km from the target launch site, any missile launch attempt with a delay of more than approximately 32 s will fail.

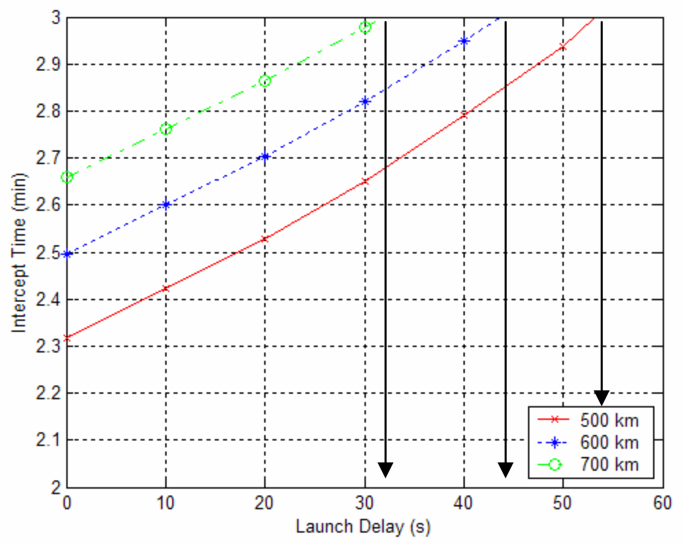


Figure 3–16. Limitation to the Tolerable Launch Delay for GM–1 Located at Attack Direction.

Returning to the San Francisco attack, it is possible to investigate the limitations in terms of location and launch delay. For GM–3, the effect of missile location can be illustrated as shown in Fig. 3–17. To accomplish a boost–phase intercept with GM–3, an attack targeting San Francisco, California requires a missile location of less than 992 km to the east of the target launch site

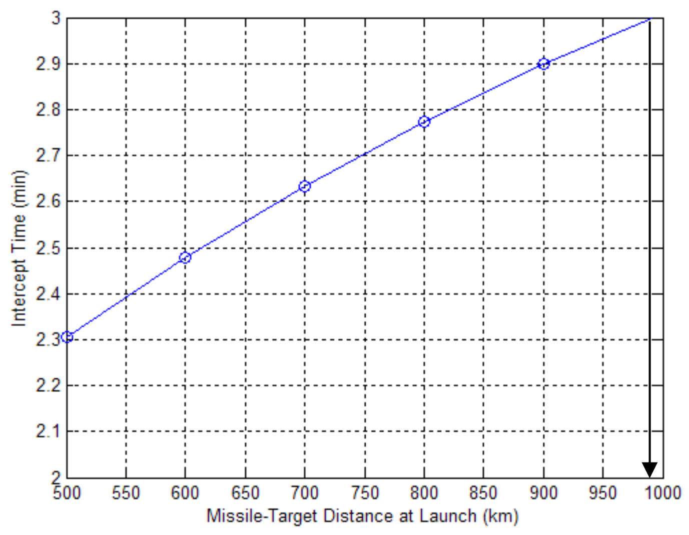


Figure 3–17. Limitation to the Missile Launch Site Distance from the Target Launch Site: 40° Angular Error, GM–3, San Francisco Attack.

Figure 3–18 shows the tolerable launch delay as a function of the missile–to–target distance at launch. For this specific scenario, it is easily possible to calculate the maximum tolerable launch delay for a given distance to the launch site. For example, assume a deployed cruiser carrying the missile to the Sea of Japan at a location 600 km east of target launch site. By using Fig. 3–18, it is possible to calculate that the missile must be launched within approximately 31 s following the target launch.

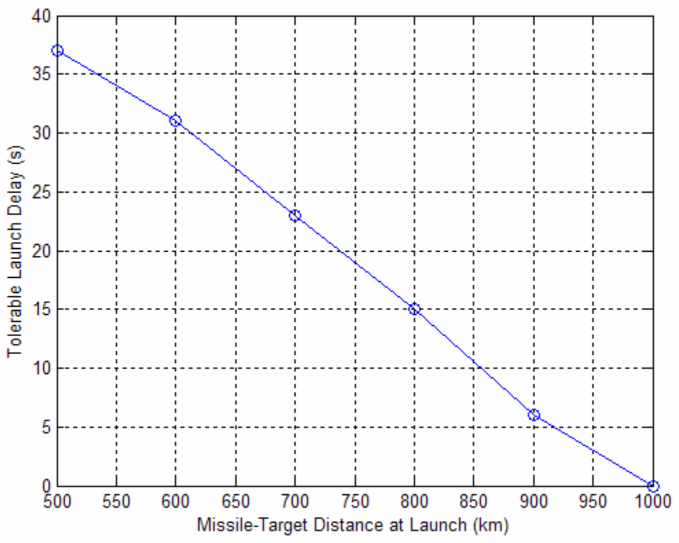


Figure 3–18. Limitation to the Tolerable Launch Delay: 40° Angular Error, GM–3, San Francisco Attack.

The investigation of missile requirements yielded the following results. Given a suitable position (in angle and distance) and launch angles, all potential missiles defined in Table 3-2 accomplished the boost–phase intercept within reasonable miss–distance and lateral divert values. The capability of the missile became important when position and launch delay deviations were introduced. Generally, the more capable the missile, the more tolerable it is to less than ideal circumstances. The positional advantage was the best when the missile was directly in the attack direction and with a zero launch delay. As the deviations from the ideal were introduced, location and launch delay tolerances decayed quickly. It was shown that, given an angular deviation and/or acceptable launch delay, the maximum distance that the missile can be located could be estimated by using the simulation.

F. SUMMARY

This chapter developed a multi–stage, boosting missile capable of intercepting a realistic target model developed in Chapter II. The proportional navigation in 3D was implemented. Test runs showed that the proportional navigation algorithm worked satisfactorily against constant–speed and realistic targets. Also developed was a non–zero–lag model defined by a 3rd order transfer function. The system lag against the constant–speed target model was tested and confirmed the theory. The missile requirements were briefly investigated in terms of capability and position. The effects of distance and angular deviations as well as the launch delay were demonstrated. This concluded the development of the target–missile model, which is the basis for the work in the following chapters. So far, physical characteristics of the target and the missile motion were examined. The next step is to determine target characteristics that affect the sensors' detection and processing capability.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RADAR CROSS SECTION AND IR ENERGY RADIATION PREDICTION

In the first part of this chapter, the monostatic radar cross section (RCS) of a three–stage generic intercontinental ballistic missile (ICBM) is predicted. This is accomplished by modeling the physical shape of each stage by using facets. A software program was then used to calculate the results for different aspect angles. All separate stages of the target are modeled to quantify the discontinuities between stages. To investigate the effect of frequency, the RCS is predicted for L–Band (1.5 GHz), S–Band (3 GHz), C–Band (6 GHz), and X–Band (10 GHz).

The investigation of the monostatic RCS is crucial since the accuracy of the RF sensor track is directly proportional to the backscatter characteristics of the target. For the target model, stage dimensions of the U.S. Peacekeeper missile are used. This missile was selected because of its long–range capability, lack of fins and protruding surfaces that would increase the RCS, and finally, the availability of detailed dimensions and other physical data in the open literature.

A common approach to predict the RCS of a three–dimensional complex target is the physical optics (PO) approximation. Many software packages provide accurate RCS results with small structures and/or low frequencies. However, while working with large structures, such as ICBMs and high frequencies, most methods result in unreasonable computational requirements due to small wavelengths, and in turn, the requirement for an extra fine mesh structure. Obtaining accurate results for electrically large structures may take months of computation. The PO method overcomes the excessive computational requirements while working with electrically large structures. There are trade–offs, however, while working with PO approximation. This method is only accurate in the specular direction, and surface waves, multiple reflections and edge diffractions are not included.

The PO software application used in this chapter is POFACETS [Ref. 14], which was developed at the Naval Postgraduate School. POFACETS provides a tool that allows the modeling of an arbitrary three–dimensional object, composed of triangular facets, visualization of its geometry, and calculation of its RCS.

In the second part of this chapter, the infrared (IR) radiation of the target plume is estimated.

A. TARGET STRUCTURE

Basic dimensions of the U.S. Peacekeeper ICBM are used for the model. Normally, the Peacekeeper has a post–boost propulsion system (PBPS), which guides it during the mid–course. However, the generic missile was constructed by combining PBPS and the payload. Figure 4–1 illustrates the details of the geometrical parts from which the model was designed. The model is composed of simple shapes such as cylinders and cones. This ICBM has a nearly smooth surface and does not have any fins or protruding surfaces.

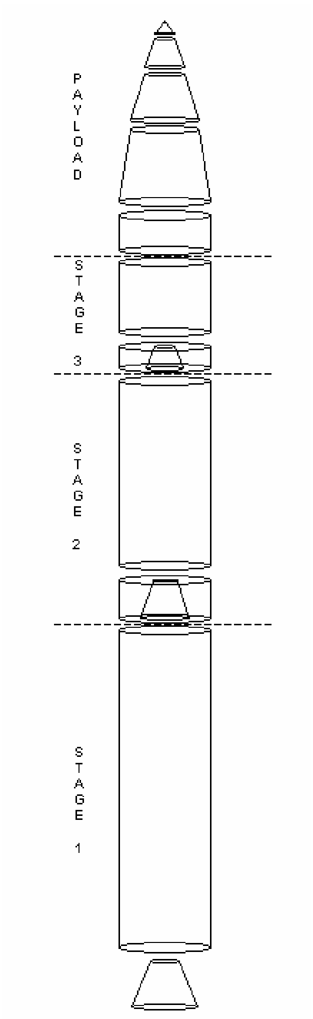
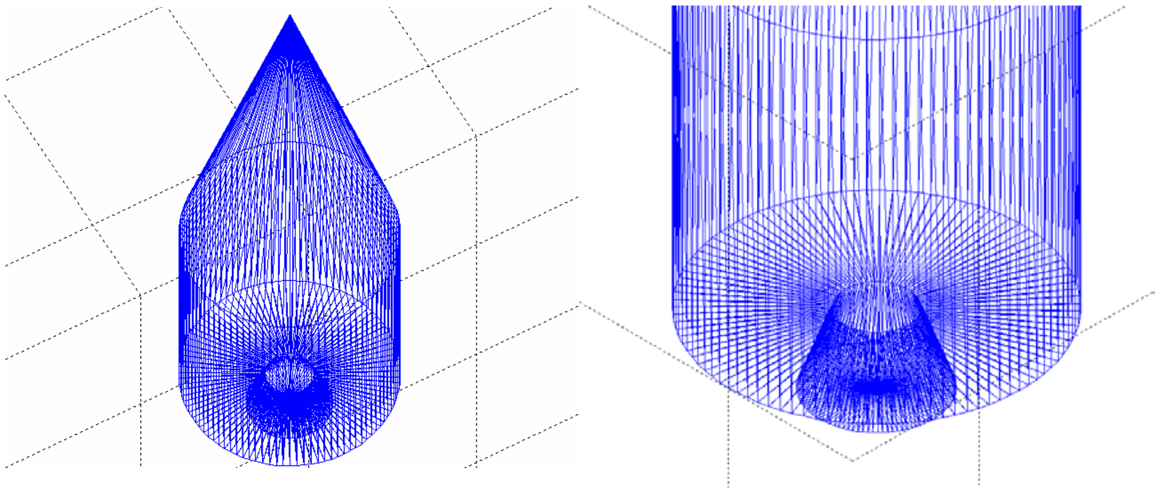


Figure 4–1. Simple Geometrical Shapes Used to Construct the Model.

The model is constructed using the original dimensions. The target is assumed to be composed of an aluminum titanium alloy. Conductivity of most of the metals varies from 1x10^7 S/m (Iron) to 6x10^7 S/m (Silver). Conductivity of pure aluminum is 3.5x10^7 S/m. For this model, conductivity was assumed to be 2x10^7 S/m. The standard deviation of the surface roughness is set to 0.3 mm.

B. POFACETS MODELING

By using the target physical dimensions, the structure is modeled in POFACETS. A MATLAB code was written to generate the coordinates of 400 points and configurations of 800 triangular facets for each stage. The main structures (cylinders) are modeled as having 100 sides. Figure 4–2 illustrates the different parts of the target emphasizing the level of detail in the construction of the model.



(a) (b)

Figure 4–2. Facet Structure Used to Construct the Model: (a) Detail, Top View and Nose Cone, (b) Detail, Nozzle.

Figure 4–3 illustrates the full–scale models of all stages. The figure emphasizes that the structures were not scaled down.

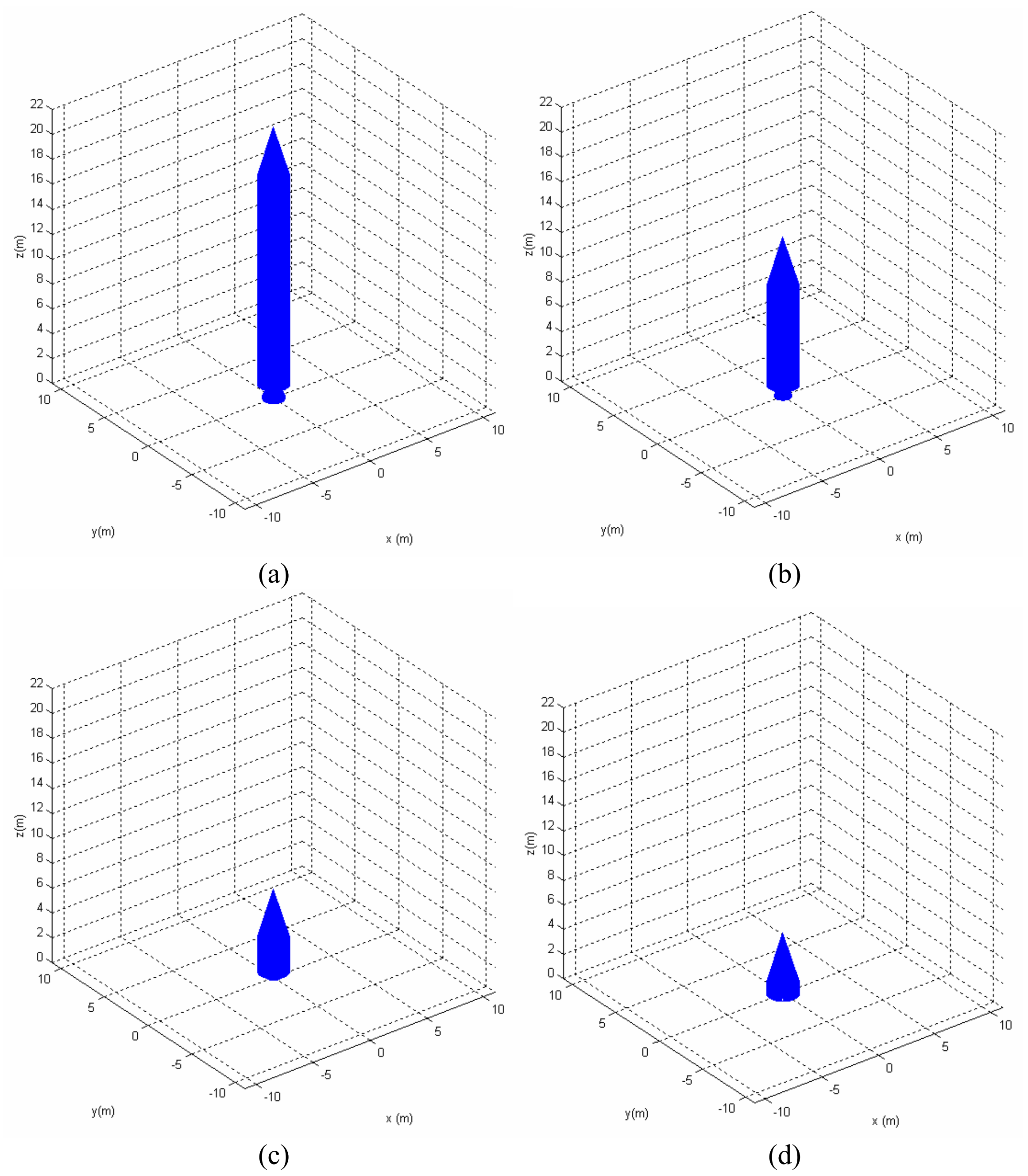


Figure 4–3. Full–Scale Models of Stages: (a) Stage–1, (b) Stage–2, (c) Stage–3, (d) Payload.

C. RCS PREDICTION

After modeling the structures by using facets, the code was run for each stage and different frequencies. The step increment for the monostatic angle θ is 1°. Figure 4–4 pictorially defines the monostatic angle θ.

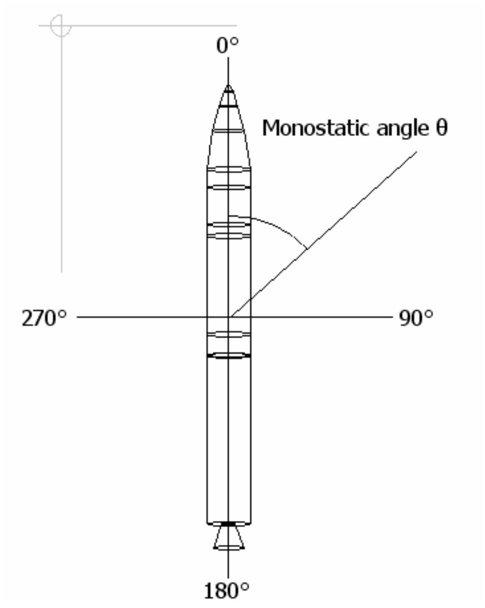
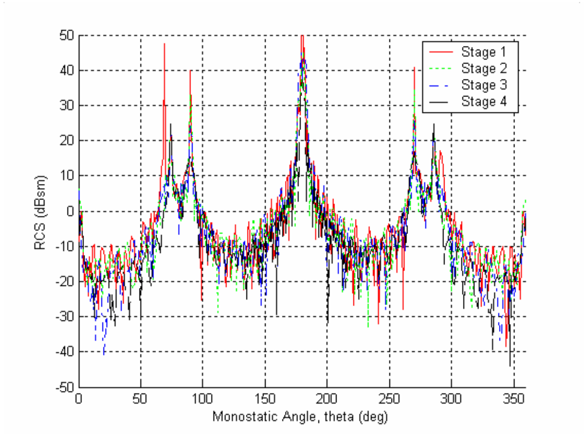
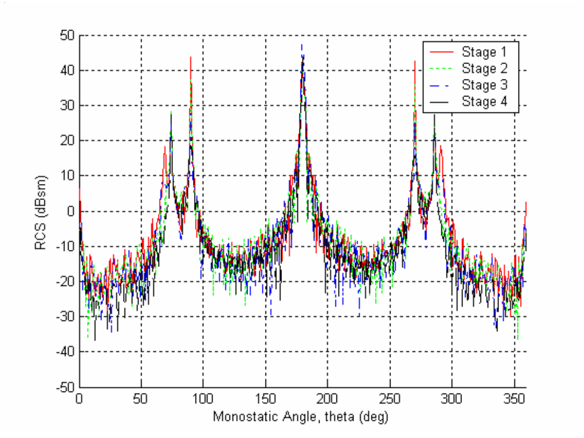


Figure 4–4. The Monostatic Angle θ.

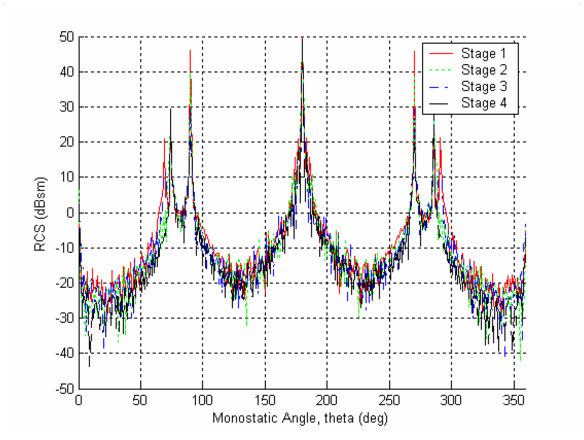
Figure 4–5 compares the monostatic RCS for each stage for the different frequencies. As Fig. 4-5 illustrates, top aspects ($\theta = 0°$) have a very low RCS value due to scattering by the nose cone in all directions other than the monostatic direction. As the aspect angle increases from 0°, the first peak occurs at approximately 75° where the slant nose cone causes a specular backscatter. The next peak takes place at $\theta = 90°$, which is perpendicular to the main fuselage. Between 90° and 160°, the RCS is between -10 and -25 dBsm. The maximum RCS occurs at 180° (bottom aspect). Since the structure is symmetric, the RCS changes symmetrically between 180° and 360°. The target stage, however, does not affect the RCS value significantly. It is very hard to discriminate the various stages from each other. Fluctuations in aspect angle within the same stage are much more significant than those between different stages.



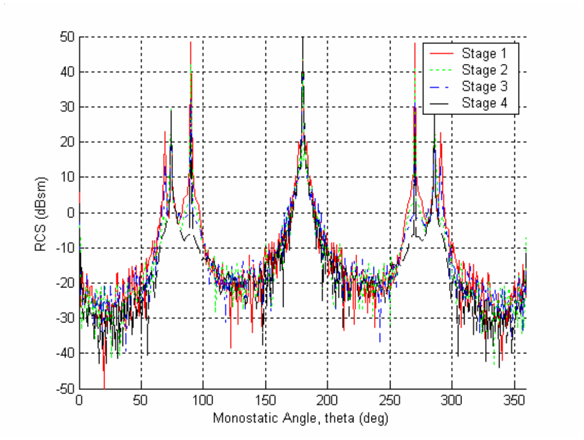
(a)



(b)



(c)



(d)

Figure 4–5. RCS Comparison of Stages: (a) L–Band (f = 1.5GHz), (b) S–Band (f = 3GHz), (c) C–Band (f = 6GHz), (d) X–Band (f = 10GHz).

The dynamic range in RCS as the aspect angle changes is enormous. Depending on the frequency, the RCS may change by as much as 100 dBsm. The smooth and non–complex structure of the target causes very low monostatic RCS at aspects other than the side ($\theta = 90°$) or the bottom ($\theta = 180°$). The simplicity of the target structure impacts the RCS significantly.

Figure 4–6 illustrates the change in RCS due to an increase in frequency for different stages. Comparison of frequencies within the same stage yields similar results. The change in RCS due to aspect angle is much more significant than the change due to frequency. As the frequency increases, maximum RCS values increase and minimum RCS values decrease, so high frequencies have a larger dynamic range.

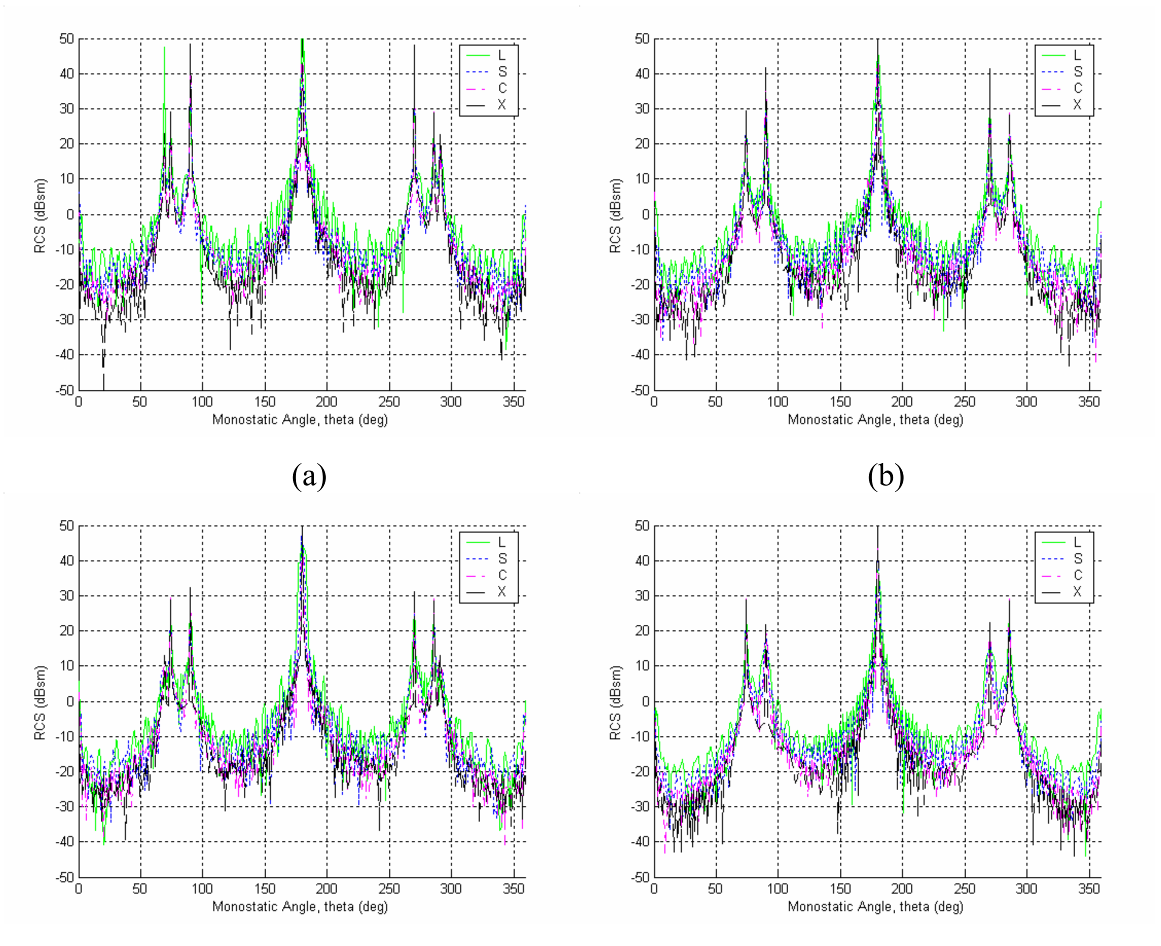


Figure 4–6. RCS Comparison of Frequencies: (a) Stage–1, (b) Stage–2, (c) Stage–3, (d) Payload.

The aspect angle between the target and the RF sensor significantly affects the RCS value. RCS can be improved by looking at the target from specular directions, such as the side or bottom. However, sensor locations have other constraints, such as territorial issues, which limit the freedom to locate the RF sensors.

Based on these results, the power requirements while designing RF sensors should consider a backscatter value on the order of -10 to -20 dBsm.

D. ESTIMATION OF PLUME IR RADIATION

The plume is the primary source of radiation for the IR sensors. One of the most encouraging features of the boost–phase ballistic missile intercept scenario is the large amount of IR radiation caused by the rocket engines of the target. Depending on the nature of the propellant used in the rocket engines (solid/liquid), ICBMs may contain water vapor, carbon dioxide and solid particulates and have temperatures of approximately 2,000K in their plumes [Ref. 15:p. 24]. Davis and Lisowski state that typical plume temperatures can be in the range of 1,500–2,000K [Ref. 5]. At lower altitudes, the energy emitted by the plume is mostly absorbed by the water vapor and the carbon dioxide in the atmosphere. As the target rises through the exoatmospheric region, the atmospheric transmission is improved [Ref. 15:p. 32].

Table 4–1 lists the assumptions are made to obtain a rough estimate of the radiation intensity of the first stage of an ICBM [Ref. 15:p. 100].

Plume Temperature at Nozzle Exit	1,800K
Average Temperature of Visible Plume	1,400K
Plume Surface Area	$600\,\text{m}^2$
Radiation Type	Isotropic

Table 4–1. ICBM Plume Parameters.

If the majority of the propellant is composed of particulates, the plume can be assumed to be a blackbody [Ref. 5]. Assuming that the plume is a blackbody at 1400K, it is possible to calculate the spectral radiant exitance $W(\lambda)$ (in $\text{W}/(\text{cm}^2 \cdot \mu\text{m})$) by using Planck's Law [Ref. 16:p. 199] as

$$W(\lambda) = \frac{C_1}{\lambda^5(\exp(C_2/\lambda T) - 1)} \tag{4-1}$$

where C_1 is 3.741×10^4 $\text{W}\cdot\text{cm}^{-2}\cdot\mu\text{m}^4$, C_2 is 1.438×10^4 $\mu\text{m}\cdot\text{K}$, λ is the wavelength in μm, and T is temperature in K.

For the above case, spectral radiant exitance can be plotted for the blackbody as shown on Fig. 4–7. A peak radiant exitance of approximately 7 $\text{W}/(\text{cm}^2\cdot\mu\text{m})$ occurs at a wavelength of $2.1\,\mu$m. However, a sensor centered at $2.1\,\mu$m is not an optimal solution for this kind of application. In a scenario where the space–based IR sensors detect and track ICBMs in the boost phase, the plume emission competes with the background clutter. Background clutter can be generated by *solar reflection*, which dominates below 3 μm; however, above 3 μm, the solar reflection is negligible [Ref. 16: p. 209]. In the daylight, the detection and tracking of ICBM plumes cannot be accomplished in the visible (VIS) or the near IR (NIR) band due to the solar reflection, even though these bands may seem to be optimal for blackbodies with temperatures of more than 1,000K [Ref. 5]. Background clutter can also be generated by the *Earth's radiation* and is significant above 5 μm [Ref. 16:p. 210] and decreases gradually below this value. Considering the regions where background clutter is dominate, the atmospheric transmittance window between 3 μm and 5 μm (mid–wave IR) is a good choice for the detection of the ICBM plume.

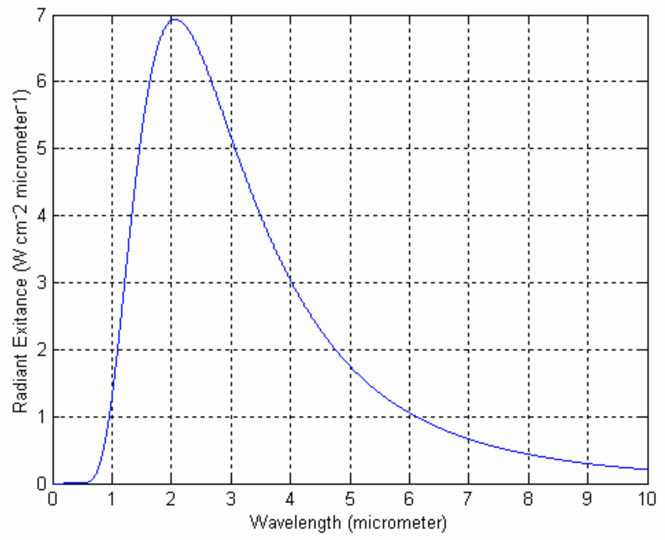


Figure 4–7. Spectral Radiant Exitance, Blackbody at 1400K.

The total radiant exitance in the 3–5 μm region can be calculated by integrating the spectral radiant exitance curve between the two wavelengths. Total radiant exitance can be calculated by using [Ref. 16:p. 206]

$$W = \int_{\lambda_1}^{\lambda_2} \varepsilon(\lambda) W_\lambda \, d\lambda \tag{4-2}$$

where W is the total radiant exitance in the given wavelength range (in $\mathrm{W/cm^2}$), $\varepsilon(\lambda)$ is the emissivity (in this case assumed to be 1), and W_λ is the spectral radiant exitance (in $\mathrm{W/(cm^2 \cdot \mu m)}$). Equation (4-2) yields a radiant exitance of $6.37\,\mathrm{W/cm^2}$. In this case, the total radiant flux for the $600\,\mathrm{m^2}$ target is calculated to be 38.22 MW. Assuming that the plume is an isotropic source of radiation, the radiation intensity is approximately 3 MW/sr [Ref. 15:p. 100].

The exact determination of an ICBM's IR signature is a complicated problem. The viewing aspect of the plume determines the power collected by the sensor. Also, consecutive stages of an ICBM have less thrust and smaller radiation intensity [Ref. 15:p. 24]. From this point, the first stage plume is assumed to be an isotropic source with a radiation intensity of 3 MW/sr. As the stages progress, the radiation intensity is assumed to be reduced in proportion to the change in fuel consumption between stages. The radiation intensity profile of the target in the intercept scenario of interest is plotted in Fig. 4–8.

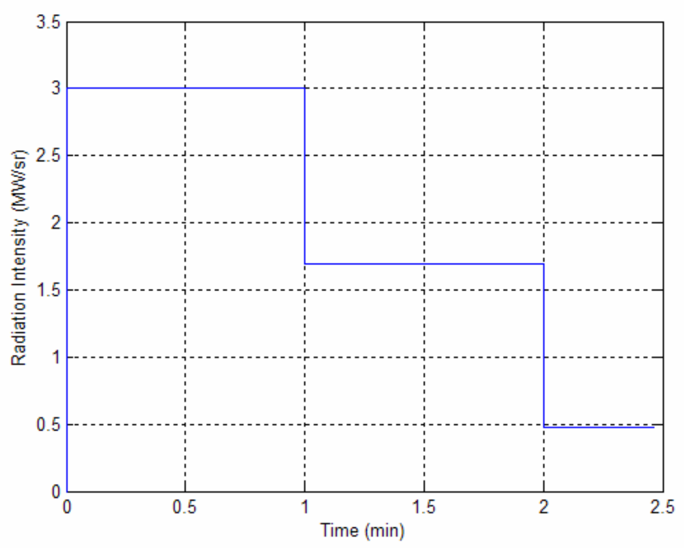


Figure 4–8. Radiation Intensity versus Time.

E. SUMMARY

This chapter investigated the target parameters from the sensors' point of view. The sensors use either radiated or reflected energy from the target in order to establish and maintain the track. The collected track information is used to construct the target position data to guide the missile to intercept. Since a successful intercept cannot be accomplished without an accurate target track, investigation of target parameters affecting the track quality is crucial. The next step is to examine the sensors.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SENSOR MODELING

The objective of this chapter is to investigate the sensor issues affecting the guidance of the missile in a boost–phase ballistic missile intercept scenario. The target position is provided to the interceptor by fusing four different sets of sensor measurements received from two ground–based radars and two LEO IR sensors. The interceptor missile is assumed to know its position by using onboard sensors, such as inertial navigation system (INS) and global positioning system (GPS). Upon receiving the target position, the missile guidance computer calculates the LOS rate and closing velocity to generate the lateral acceleration commands in accordance with the proportional navigation guidance law.

The miss distance will always be negligible given the following conditions:

- The missile can be launched with an acceptable initial heading error.
- The received target position is accurate.
- The target position can be received and lateral acceleration commands can be applied with zero delay.

In reality, however, these conditions cannot be accomplished due to the transmission delay and the tracking inaccuracies. This chapter will examine the conditions generating the miss distance.

Figure 5–1 shows a geographic scenario summarizing the boost–phase ballistic missile intercept investigated. The missile is located 600 km east of the target launch site. RF–1 and RF–2 are located at bearings of 125° and 135° and distances of 600 km, respectively. The IR sensors are located over the target and missile launch sites.

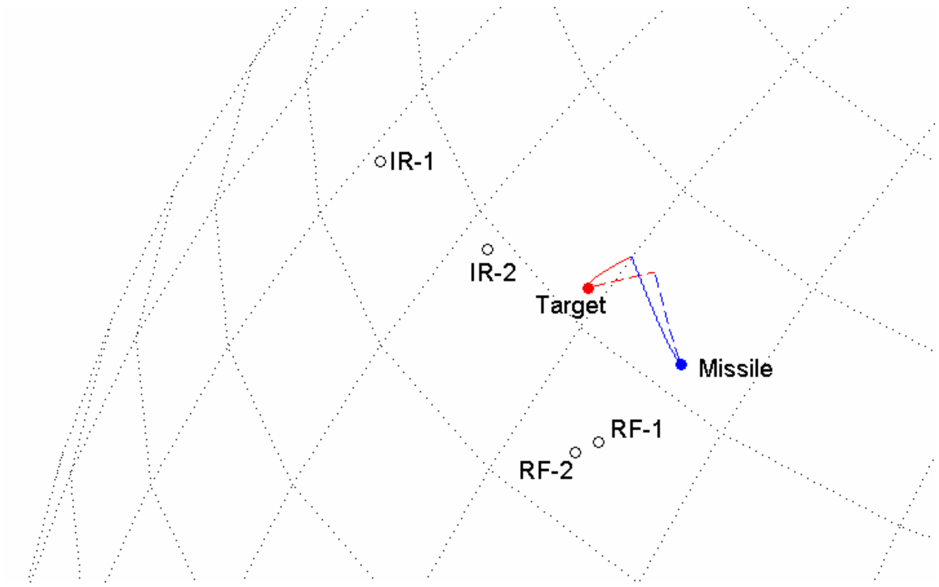


Figure 5–1. The Geographic Scenario for the Boost–phase Ballistic Missile Intercept including Locations of the Sensors, the Missile, and the Target.

Figure 5–2 illustrates the schematic drawing of the scenario. We consider that the target is launched from a selected missile site in North Korea toward San Francisco. The interceptor missile is launched after a given delay following the target launch from the Sea of Japan. Two RF sensors and two IR sensors track the target. Tracking of the target refers to providing the target position in angle and range by the RF sensors and, target position in angle by the IR sensors within given update intervals. The data fusion center combines the target position inputs by using an algorithm in order to generate the target position. This position data is transmitted to the missile, which is guided to accomplish the intercept. The data link refers to the channels through which target position data flows from the sensors to the fusion center, and from fusion center to the missile.

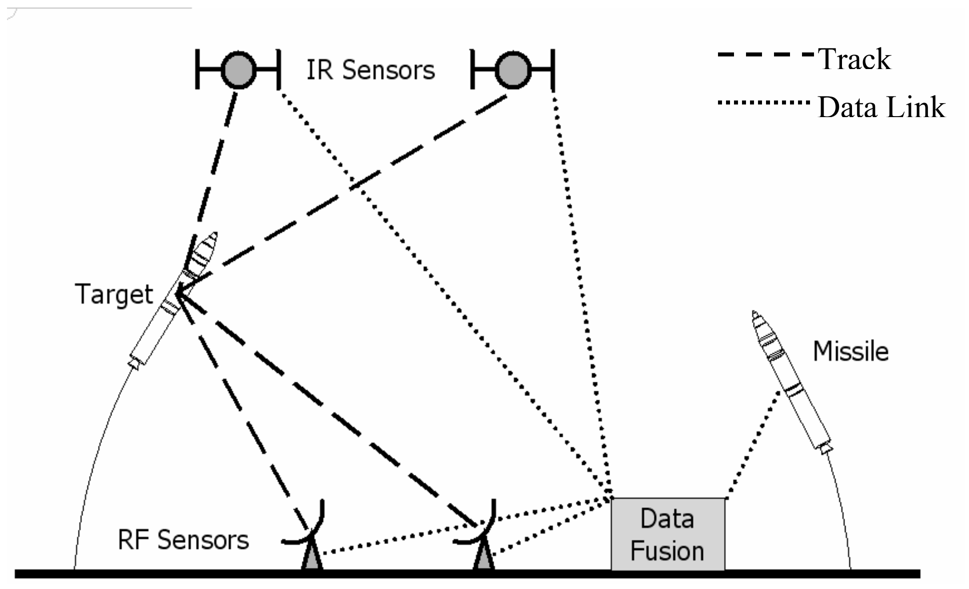


Figure 5–2. The Schematic Scenario for the Boost–phase Ballistic Missile Intercept including Locations of the Sensors, the Missile, and the Target.

The RF sensor locations are chosen by using the simulation to estimate an average RCS for several possible RF sensor locations for X-band. The possible locations are shown in Fig. 5–3. The average RCS value between the observation points are determined through interpolation. Due to territorial limitations, RF sensor locations are limited to the eastern sector of the target launch site (i.e., 0° to 180° bearing). The average RCS is measured for distances in the range of 300 to 1,000 km from the target launch site.

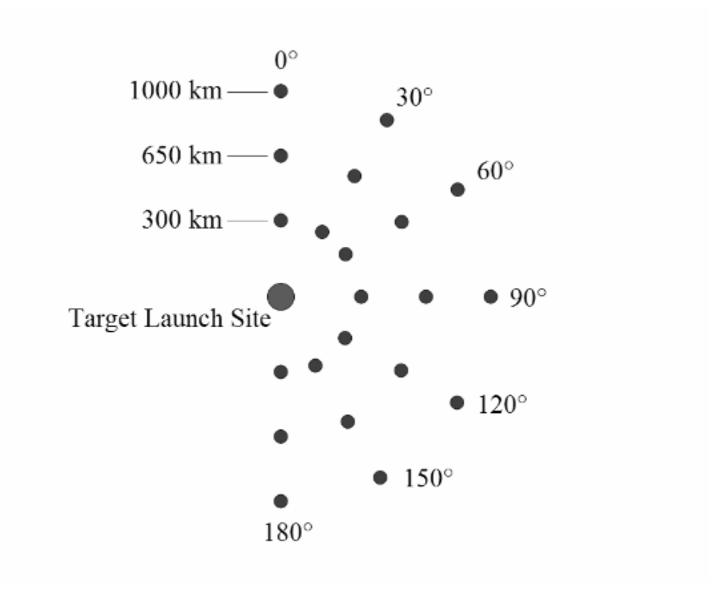


Figure 5–3. RCS Sampling Locations.

Figure 5–4 illustrates the average RCS measured during intercept as a function of bearing and range to the target. This provides good insight about the characteristics of the monostatic backscatter from the target. The bearing is measured from true north. In order to reference the bearing measurement to the attack direction, the test case attack direction of 50º should be subtracted from the bearing values shown in Fig. 5–4. Tracking accuracy of the RF sensor is a function of the SNR, which is proportional to the RCS and inversely proportional to the fourth power of range. The investigation for optimization of the RF sensor location yielded approximately 125° in bearing measured from true north or 75° measured from the target attack direction. Although the RCS increases as the distance increases at the optimal bearing, the reduction in the SNR due to range is much more significant. Thus, the optimal range for the RF sensor is always the minimum possible range.

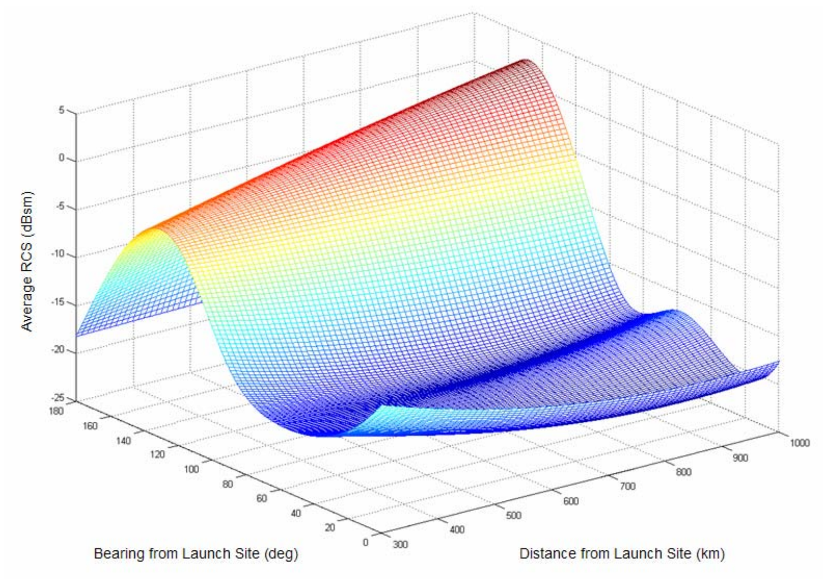


Figure 5–4. Average RCS Seen by RF Sensor as a Function of Bearing and Range from the Target Launch Site.

A. TRANSMISSION DELAY

A transmission delay occurs when measured target data are transmitted from the sensors to the fusion processor and from the fusion processor to the missile. Processing delays in sensors and the fusion processor can also be included in this delay. Computing the exact value of the transmission delay incurred is not within the scope of this research.

Transmission delay can be modeled by assuming that the received target data by the interceptor reflect an instance in the past of the target. As a result, the missile always lags in the collision geometry since the acceleration commands are generated to intercept a point in the trail of the target.

Based on multiple simulation runs, the miss distance d_m (in m) as a function of the transmission delay τ_t (in s) was determined to be

$$d_m = 3.3 \times 10^3 \tau_t + 5.5 \qquad (5\text{-}1)$$

Equation (5-1) implies that the transmission delay introduces a miss distance as a function of the traveled distance by the target during the time period in which delay occurs. It should also be noted that this is an approximate result since the intercept geometry affects the magnitude of this miss distance.

B. TRACKING INACCURACIES

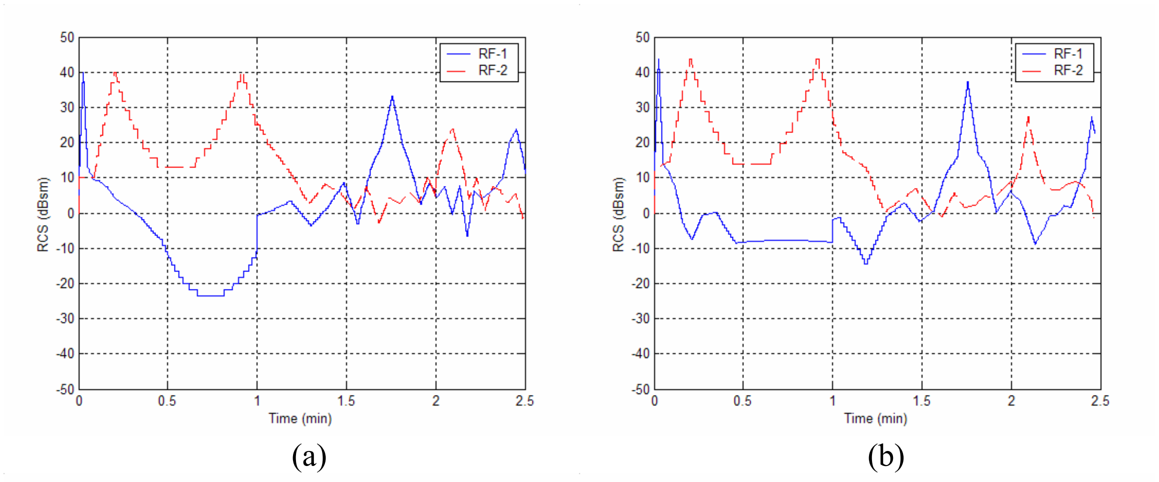
The major factor affecting the RF sensor tracking accuracy is SNR that depends on peak power, antenna gain, pulsewidth, number of integrated pulses, radar cross section, range to target, and thermal noise power. IR sensor accuracy is affected by instantaneous field of view (IFOV) of the detectors and range to target.

It is assumed that the sensors are tracking the target during intercept. In other words, the target acquisition phase is not included in the model.

1. RF Sensor Inaccuracies

Given certain radar parameters, there are two factors continuously changing during the flight, RCS and the range to target. The RCS seen by two separate RF sensors dynamically change depending on the target aspect with respect to the sensor position and the stage of the target. In order to quantify the continuously changing RCS within the intercept, the results concerning RCS in Chapter IV are collected in 20 .mat files. The target and missile models developed in Chapters II and III were used to quantify the radar cross–section during the boost phase intercept. Two generic RF sensors southeast of the target launch site in the Sea of Japan were defined. One of the RF sensors (RF–1) is located at N 37°46 E 134°35. The other RF sensor (RF–2) is located at N 37°05 E 133°46. The RF sensors are located at 600 km in range and, 125° and 135° in bearing, respectively, relative to the target launch site.

The simulation was run several times to examine the RCS seen by the RF sensors. To accomplish this target and missile were allowed to fly until the interception takes place. The RCS data predicted in Chapter IV was written into lookup tables and interpolated to a precision of 0.1°. For each step time of the simulation, target aspect angles seen by both RF sensors were computed and rounded to the multiples of 0.1°. By using the computed aspect angle and the lookup tables, the RCS value was determined. Figure 5–5 summarizes the change in the RCS due to aspect angle and stage change as interception progresses.



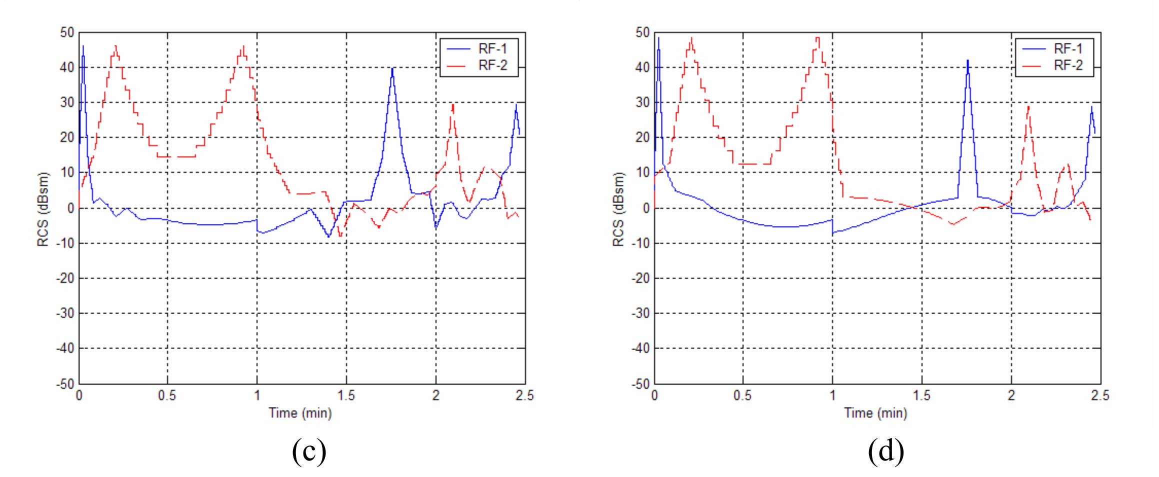


Figure 5–5. RCS Seen by RF–1 and RF–2 during the Intercept: (a) L–Band, (b) S–Band, (c) C–Band, (d) X–Band.

The stair step structure observed in the figures is a result of precision of the lookup tables. By examining Fig. 5–5, it is possible to conclude that the RCS fluctuates between –10 dBsm and +50 dBsm as the aspect angle changes during the flight. Also note the compensating nature of the two sensor positions. The locations of RF sensors were specifically selected to enhance the overall SNR after fusion, that is, when one sensor sees a low RCS, the other sees a high RCS avoiding poor track quality. X–Band (10 GHz) radars were used to generate the rest of the results.

The RF sensor–to–target range during intercept can also be calculated. Fig. 5–6 illustrates the sensor–to–target range for each RF sensor. The nearly constant sensor to target range shown in Fig. 5–6 indicates that the sensors are located so that they see a monostatic angle around 90° during most of the intercept. This causes a specular backscatter and a better average RCS.

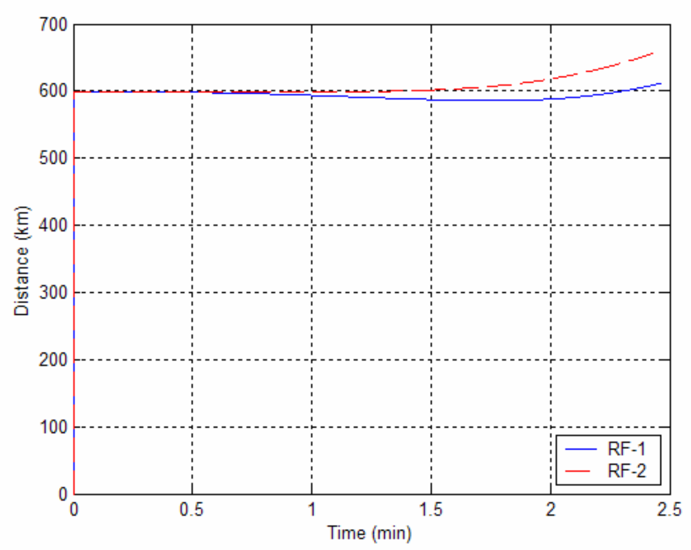


Figure 5–6. RF Sensor to Target Range.

Given the above scenario, it is also possible to determine some RF sensor requirements. The RF sensor–to–target range estimation shows that the RF sensors should have a minimum unambiguous range of $R_{un} = 1{,}000$ km. A low pulse repetition frequency (LPRF) is used. To calculate the maximum PRF f_p, [Ref. 17: p. 3] is used

$$f_p = \frac{c}{2R_{un}} \tag{5-2}$$

where c is the speed of light (in m/s). With $R_{un} = 1{,}000$ km, the maximum PRF is $f_p = 150$ Hz. For the antenna, a pencil–beam is used. In practice, the gain G of the antenna is approximated by [Ref. 18: p. 298]

$$G = \frac{26000}{\theta_{3dB}^{a}\theta_{3dB}^{e}} \tag{5-3}$$

where θ_{3dB}^{a} is the half–power (3–dB) beamwidth in the azimuth direction (in degrees), and θ_{3dB}^{e} is the half–power (3–dB) beamwidth in the elevation direction (in degrees). Assuming that the antenna is a parabolic reflector, the antenna's physical area A_p is solved using [Ref. 18:p. 17]

$$A_p = \frac{G\lambda^2}{4\pi\varepsilon_{ap}} \tag{5-4}$$

where ε_{ap} is the aperture efficiency and λ is the wavelength (in m).

A possible set of half–power beamwidths yields the gains and antenna diameters shown in Table 5–1. Aperture efficiency is assumed to be 0.55. Table 5–1 shows that antenna half power beamwidth as small as 0.5º result in reasonable antenna diameters for surface based systems. By using (5-4), it is possible to calculate that antenna half–power beamwidths smaller than the values given in Table 5–1, which result in excessive antenna diameters that may be infeasible to design and operate.

Half Power Beamwidth (Degrees)	**Gain (dB)**	**Antenna Diameter (m)**
0.5×0.5	50	4.15
1×1	44	2.08

Table 5–1. Gain and Antenna Diameter versus Required Half Power Beamwidth.

The range of parameters shown in Table 5–2 will be examined for the RF sensors. The RF sensor frequency is a given parameter in this research. Antenna beamwidth and gain are calculated according to the antenna size requirements. PRF is calculated as a function of range certainty requirements. The peak power, pulsewidth and pulse integration and noise factor values are set in the range of typical values for classical radar systems.

Parameter	**Value**
Frequency, c/λ	10 GHz (X–Band)
Peak Power, P_t	100 kW to 1 MW
Antenna Gain, G	35 to 50 dB
Beam width, θ_{3dB}	0.5 to 1 degree
Pulse width, τ	10 to 50 μs
PRF, f_p	150 Hz
Number of Pulses Integrated, N_i	10 to 20
Receiver Noise Factor, F	4

Table 5–2. RF Sensor Parameters to be Examined.

Solving the radar range equation [Ref. 16: p. 159] for single pulse SNR $(S/N)_1$, assuming that the number of integrated pulses $N_i = 1$, the transmitter and the receiver use the same antenna, the bandwidth B is equal to the reciprocal of the pulsewidth $1/\tau$, the antenna temperature T_0 = 290K, and neglecting other losses yields

$$(S/N)_1 = \frac{P_t G^2 \tau \sigma \lambda^2}{(4\pi)^3 k T_0 F R^4} \tag{5-5}$$

where σ is the RCS (in m^2), k is the Boltzman constant (in J/K) and R is the range (in m) to target. The SNR affects the tracking quality of the RF sensors. The RMS error in angle and range due to thermal noise is given by [Ref. 16:pp. 160-163]

$$\sigma_{angle} = \frac{\theta_{3dB}}{K\sqrt{(2(S/N)_1)N_i}} \tag{5-6}$$

and

$$\sigma_{range} = \frac{c\tau}{2}\frac{1}{K\sqrt{(2(S/N)_1)N_i}} \tag{5-7}$$

The constant K for the RMS angle error is approximately 1.7 for a monopulse tracker. Also, the constant K for the RMS range error is a factor between 1 and 2. Both constants are assumed to be 1.7. Equation (5-6) suggests that the angle accuracy of the tracking radar depends on the half–power beamwidth of the antenna and the SNR, and (5-7) indicates that range accuracy depends on the pulsewidth and the SNR. To investigate the effect of the different radar parameters, one parameter is changed at a time while the others are kept constant. The parameters kept constant are $P_t = 1$ MW, θ_{3dB} = 0.5º, τ = 50 μs, and $N_i = 20$.

Variation in the peak power is examined first. As the peak power increases, the tracking accuracy improves. Figure 5–7 illustrates the RMS angle and range errors for the scenario (defined at the beginning of this chapter) as the peak power changes from 100 kW to 1 MW. The tracking accuracy strongly depends on the target RCS. When the target RCS is low, the RMS tracking error increases. Stage discontinuities are also evident in Fig. 5–7. For example, at the beginning of stage–2 (1 minute), the RMS angle error for 100 kW radar reaches a peak of 0.03°, which corresponds to an approximate distance of 314 m at the sensor–target range. Also, the range error reaches a peak of 450 m under the same circumstances.

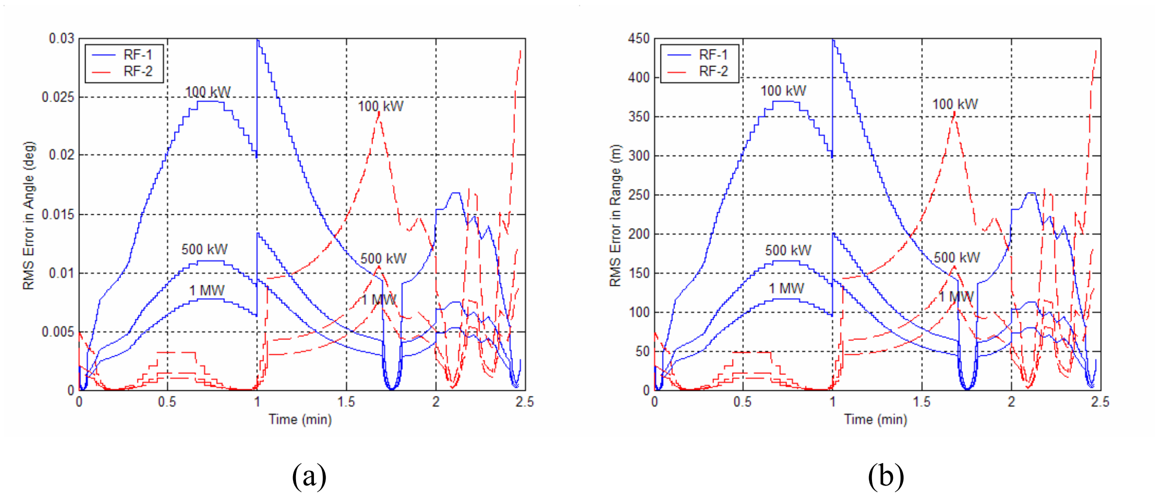
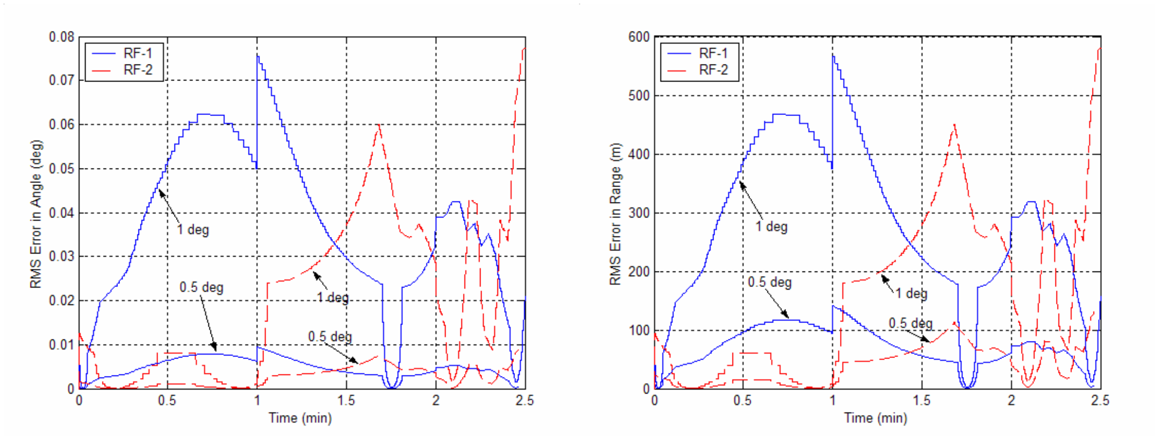


Figure 5–7. Effect of Peak Power to Tracking Accuracy: (a) Angle, (b) Range.

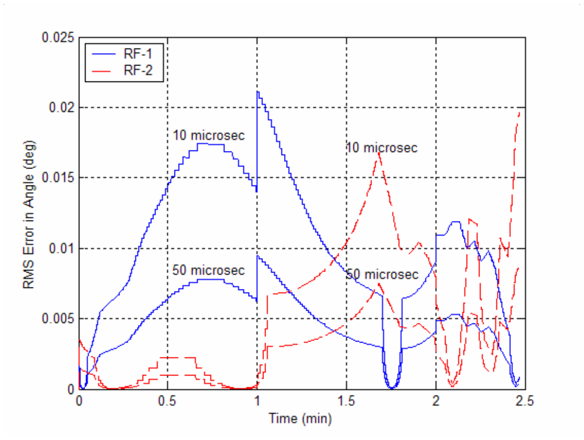
Another factor is the antenna half–power beamwidth. As the beamwidth is reduced, angular tracking accuracy is improved. However, since the antenna gain increases as the beamwidth decreases, range accuracy is also improved. Figure 5–8 illustrates the RMS angle and range errors for the given scenario as the antenna half–power beamwidth is reduced from 1×1 to 0.5×0.5 degrees. Plots of Fig. 5–8 are similar to those of Fig. 5–7 in shape but have different magnitudes. The impact of antenna half–power beamwidth is significant. For example, at the beginning of stage–2 (1 minute), a 0.5 degree increase causes the RMS angular error to increase from approximately 0.01 degrees to 0.075 degrees, which corresponds to a change of 680 m at the target range. Also, under the same circumstances, the RMS range error increases from approximately 130 m to 560 m.

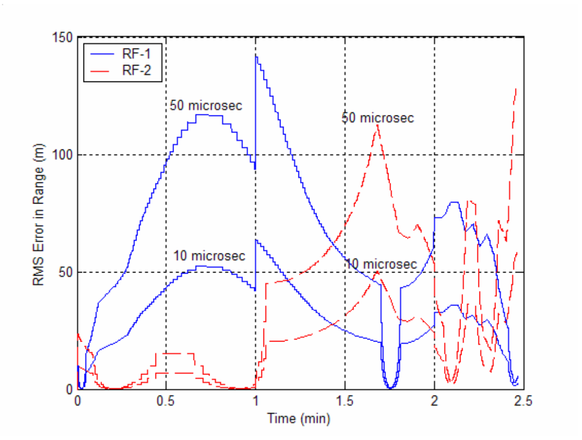


(a) (b)

Figure 5–8. Effect of Half–power Beamwidth to Tracking Accuracy: (a) Angle, (b) Range.

Another factor of interest is the pulsewidth. The pulsewidth was tested in the range of 10 μs to 50 μs . As the pulsewidth increases, the range accuracy decreases while angular accuracy is improved. The reason for the improved angular accuracy is the bandwidth. Since the bandwidth decreases as the pulsewidth increases, the SNR improves, which in turn, causes a better angular tracking capability. Figure 5–9 illustrates the RMS angle and range errors for the given scenario as the pulsewidth is changed. Note the inverse relationship between angle and range. For the time of 1 minute, the angular accuracy improves from an approximate value of 209 m to 105 m as pulsewidth is increased from 10 μs to 50 μs . However, as the pulsewidth is changed, the range accuracy decays from approximately 60 m to 140 m.

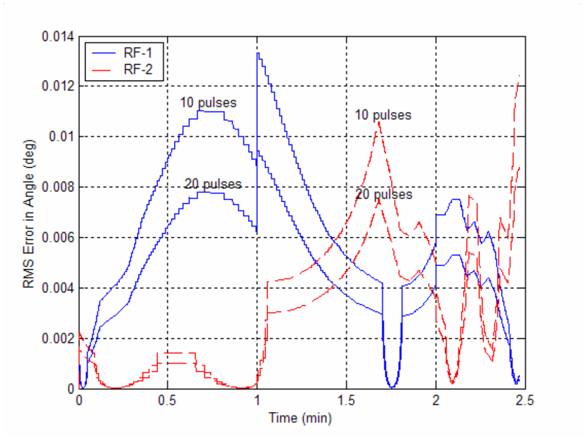


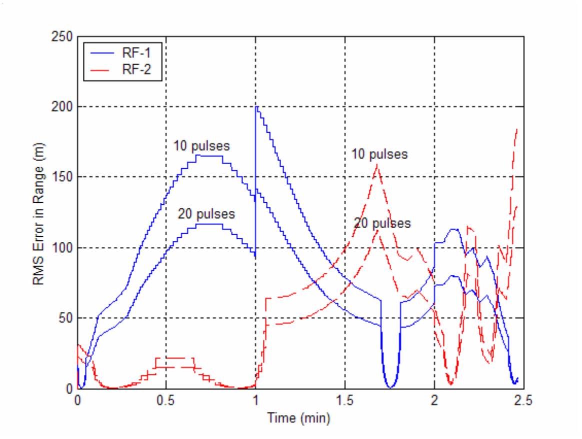


(a) (b)

Figure 5–9. Effect of Pulsewidth to Tracking Accuracy: (a) Angle, (b) Range.

The final factor investigated was the number of pulses integrated, N_i. As the number of pulses integrated is increased, tracking accuracy is improved. Figure 5–10 illustrates the RMS angle and range errors for the given scenario as the number of pulses integrated is increased from 10 to 20. For both angle and range, an increase in N_i improves tracking accuracy. For example, at the beginning of stage–2 (1 minute), an additional 10 integrated pulses cause an approximate tracking improvement of 42 m in angle at the target range and 60 m in range.





(a) (b)

Figure 5–10. Effect of Pulse Integration to Tracking Accuracy: (a) Angle, (b) Range.

Table 5–3 shows the deduced RF sensor parameters to improve the accuracy. Maximum values from the given table for peak power, pulsewidth, and number of pulses integrated are chosen while minimum of antenna half–power beamwidths are chosen.

Parameter	**Value**
Frequency	10 GHz (X–Band)
Peak Power	1 MW
Antenna Gain	50 dB
Beam width	0.5×0.5 degrees
Pulse width	50 μs
PRF	150 Hz
Number of Pulses Integrated	20
Receiver Noise Factor	4

Table 5–3. Generic Radar Parameters.

The parameters given in Table 5–3 can be used to calculate $(S/N)_1$, as shown in Fig. 5–11 using (5-5). As the figure illustrates, $(S/N)_1$ changes in the approximate range of 10 to 70 dB during the intercept. The SNR fluctuates as the intercept progresses and aspect angles change. At the initial stages, RF–1 sees a lower SNR. However, the good SNR obtained by RF–2 compensates for this reduction in tracking accuracy. Similarly, RF–1 sees good SNR at the terminal phase of the intercept while RF–2 is degraded.

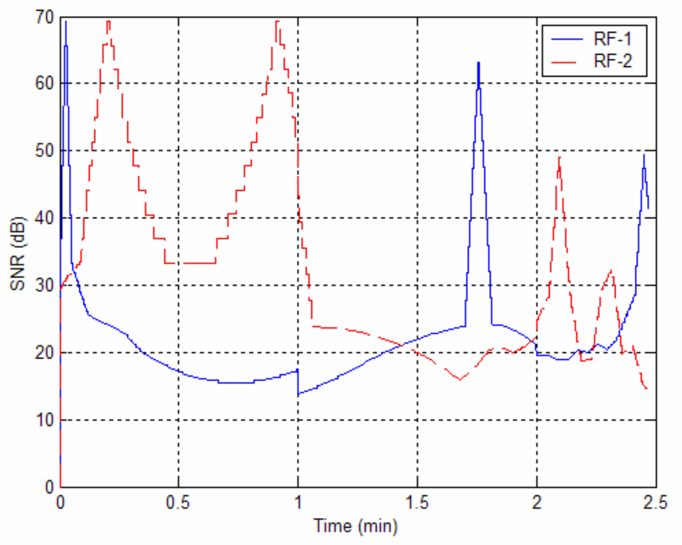
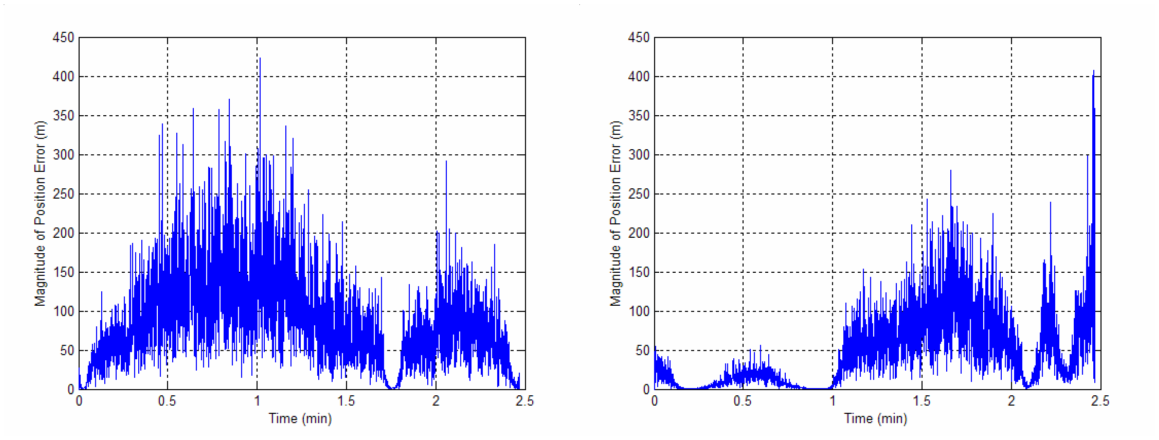


Figure 5–11. Single Pulse SNR versus Flight Time.

After calculating the RMS errors based on (5-6) and (5-7), an error magnitude was calculated for both angle and range. They are included in the simulation to quantify their effect on the sensor's determination of the target position. The following procedure is used to include the random errors:

- SNR is calculated by using (5-5).
- RMS angular error is calculated by using (5-6).
- Using the MATLAB function *normrnd*, two random angle error values are generated having a Gaussian distribution with zero mean and the standard deviation calculated in the previous step.
- RF sensor–to–target vector, which is obtained by subtracting the sensor and the target position vectors in the Cartesian coordinate system, is converted to spherical coordinates to obtain angles θ and ϕ, and range R.
- Generated random angle errors are added to θ and ϕ.
- RMS range error is calculated by using (5-7).
- Using the MATLAB function *normrnd*, a random range error value is generated having a Gaussian distribution with zero mean and the standard deviation calculated in the previous step.
- Generated random range error is added to range R.
- Resulting RF sensor–to–target vector is converted back to the Cartesian coordinate system.

By using the radar parameters given in Table 5–3, the RF sensor accuracy was quantified for the given scenario. Figure 5–12 illustrates the difference between the true position and the position sensed by the two separate sensors, RF–1 and RF–2. Note the statistical nature of the sensed target position. So far, the missile was guided assuming that the received target position reflects the true position. However, under given circumstances, Fig. 5–12 shows that the sensed target position data may be in error up to 400 m.



(a) (b)

Figure 5–12. Magnitude of Position Error versus Flight Time, (a) RF–1, (b) RF–2.

From now on, the realistic target track data for the missile guidance is used. Next, the errors caused by the IR sensors are quantified.

2. IR Sensor Inaccuracies

The IR sensors are considered as step–stare focal plane arrays (FPA) that are mounted on LEO satellites. Staring sensors are relatively new in the area of detection and tracking of ICBMs. In this concept, instead of scanning a relatively small number of detectors along the field of view, the total search field of view is covered by using a staring sensor composed of a large number of detectors. The increase in the integration time and the resulting improvement in sensitivity is one of the most important advantages of staring sensors. However, this technique may require focal plane arrays composed of millions of detectors, which is a technological challenge. The total field of view can be further improved by stepping the staring sensor between different positions, which is called the step stare approach [Ref. 15:pp. 106-107]. The FPA requirements are calculated below.

Atmospheric transmittance between the space–based sensor and the target is significant when target is at low altitude and negligible when the target is at high altitude. In order to calculate the atmospheric transmittance under given circumstances, atmospheric radiation codes, such as MODTRAN [Ref. 19] are widely used. SEARAD [Ref. 19] is a related code, which is incorporated into MODTRAN. SEARAD was used to investigate the atmospheric transmittance for the given scenario in the band of 3–5 μm . The simulation results show that for the given scenario, the missile accomplishes the intercept at an altitude of approximately 120 km. However, calculations on the target trajectory in Chapter II also reveal that this altitude may be as high as 250 km at burnout. The SEARAD is used to quantify the atmospheric transmittance for a sensor looking down vertically for various target altitudes. Figure 5–13 illustrates the average atmospheric transmittance versus target height. The output of SEARAD depends on the computations in 3–5 μm band. As shown in the figure, the atmospheric transmittance quickly improves in first 10 km of the target ascent.

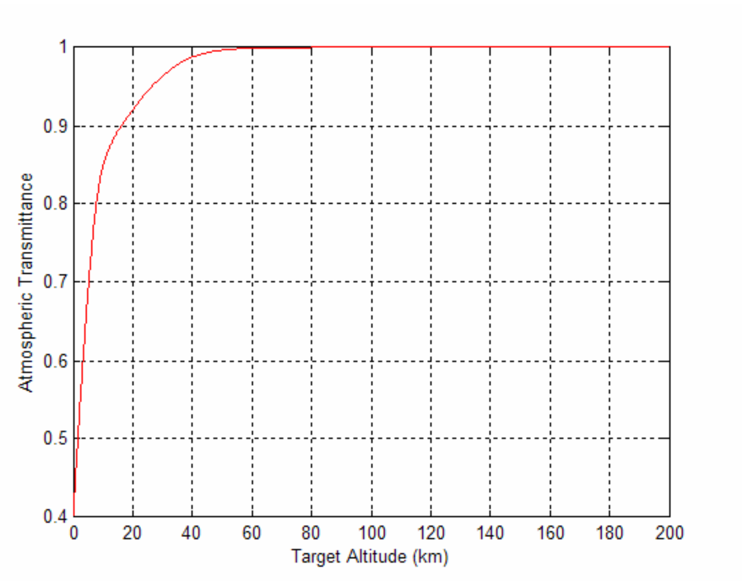


Figure 5–13. Atmospheric Transmittance versus Target Height.

Recall that the approximate radiation intensity of the plume was examined in Chapter IV. Since the sensors are of the lookdown type, the Earth's radiation (clutter) is the major system noise source. For simplicity, the SNR is neglected and the signal–to–clutter ratio is calculated for the given rocket engine plume assumed to be a blackbody on a mixed terrain clutter.

A FPA with a detector size of $20\mu m \times 20\mu m$ was considered. For an IFOV of 20 microradians, a focal length of $f = 1$ meter is required. Matching the diffraction spot size (main lobe of the Airy disk) yields an approximate optics diameter of 50 centimeters.

North Korea has an approximate area of 120,000 km^2. For example, to cover this area with 16 steps requires a coverage of 7,500 km^2 per FPA step. This yields a field of view of 0.086 radians for a sensor located at a height of 1,000 km. Knowing that IFOV is 20 microradians, a generic FPA can be designed with approximately $4{,}300 \times 4{,}300$ elements. In this design, each sensor element sees a $400\,\mathrm{m}^2$ box on the surface of the Earth.

The Earth's radiance integrated over the 3–5 μm band is given as 300×10^{-6} $\mathrm{W/(sr \cdot cm^2)}$ for a mixed terrain [Ref. 16: p. 210]. For a footprint of $400\,\mathrm{m}^2$, the radiation intensity of the clutter is 1.2 kW/sr. From Chapter IV, recall that the radiation intensity of the target is 3 MW/sr in stage–1. At the beginning of the target flight (target and clutter are at the same range), the signal–to–clutter ratio is approximately 33 dB. This is an approximate value since the decrease in the target–to–sensor range with respect to the constant ground–to–sensor distance, staging effects, and atmospheric transmittance affect the radiation intensity of the target.

The signal power S collected by the electro–optic system can be defined as [Ref. 20]

$$S = \frac{\pi D^2}{4}\left(\frac{J}{R^2}\right) \tag{5-8}$$

where D is the optics diameter, J is the radiation intensity, and R is the range between the sensor and the source of the radiation. The signal–to–clutter ratio can be plotted for the sensors by using (5-8) and the atmospheric transmittance effect given in Fig. 5–13 as shown on Fig. 5–14. The signal and clutter power are calculated in the simulation continuously. The computation yields an approximate signal–to–clutter ratio between 30 and 34 dB during the intercept. Note the effect of the atmospheric transmittance at the beginning of the intercept. The atmospheric transmittance quickly improves as the target height increases.

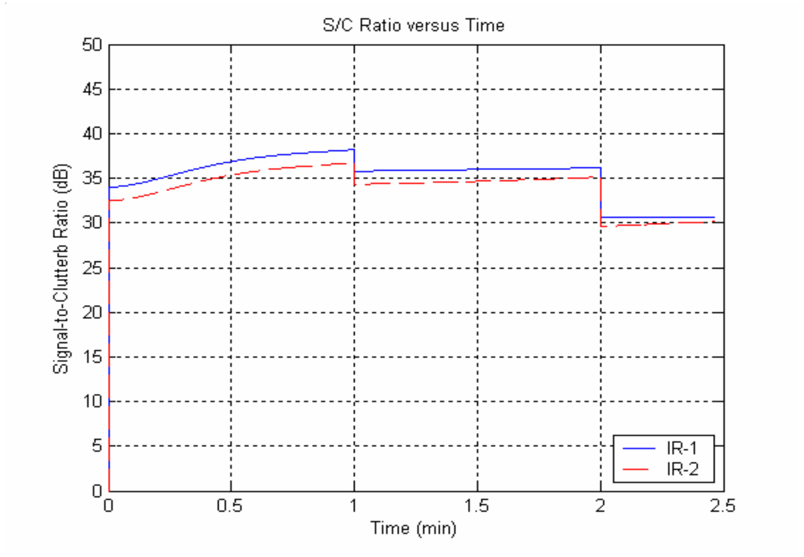


Figure 5–14. Signal–to–Clutter Ratio of IR Sensors.

The major factors affecting the IR sensor accuracy are IFOV and the sensor height. One of the IR sensors (IR–1) is located on top of the target launch site. The other IR sensor (IR–2) is located on top of the missile launch site. Table 5–4 summarizes the parameters used to model the IR sensors.

Parameter	**Value**
Type	Step–Stare FPA
Optics Diameter	50 cm
Focal Length	1 m
Sensor	HgCdTe 3–5 μm
Sensor Height	1,000 km
IFOV	20 microradians

Table 5–4. IR Sensor Parameters.

It is assumed that the exact IR sensor positions are known by means of onboard sensors such as GPS. It is also assumed that the IR sensors continuously track the target during the intercept and can provide only target angle data. Given these parameters, the target position can be deduced by using triangulation. The concept of triangulation uses the law of sines and is illustrated in Fig. 5–15 as well as the two IR sensors and the target.

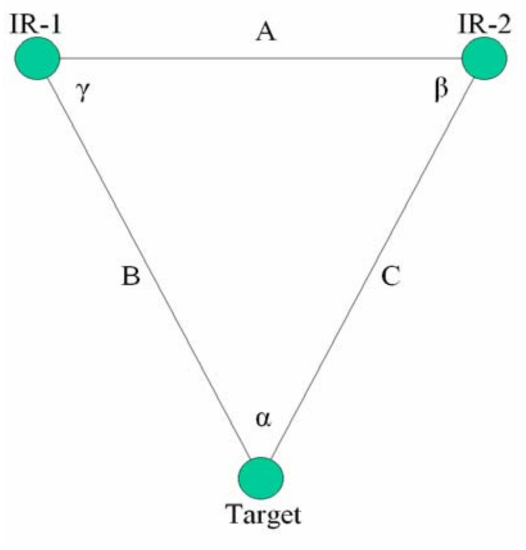


Figure 5–15. The IR sensor–Target Triangle.

The law of sines relates the angles and the distances as

$$\frac{A}{\sin\alpha}=\frac{B}{\sin\beta}=\frac{C}{\sin\gamma}. \tag{5–9}$$

Since A, β and γ are known, all other angles and distances can be calculated easily.

The size of the IFOV causes an error in angles β and γ and can generate an erroneous target position. The following procedure is used to model the IR sensor inaccuracies:

- Using the MATLAB function *rand*, two random angle error values having a uniform distribution with zero mean and limiting values of +/– IFOV/2 are generated.
- The IR sensor–to–target vector is converted to spherical coordinates to obtain angles θ and ϕ.
- The generated random angle errors are added to θ and ϕ.
- The resulting RF sensor to target vector is converted back to the Cartesian coordinate system.
- The sensed target position is deduced by using (5–9).

Figure 5–16 illustrates the resulting IR derived target position. The sensor location and specification selections result in good IR tracking accuracy. The magnitude of target position error remains under approximately 30 m during the intercept. IR sensor track data is an important component of the fused target position since the IR sensors are more resistant to electronic attack as described in Chapter VI.

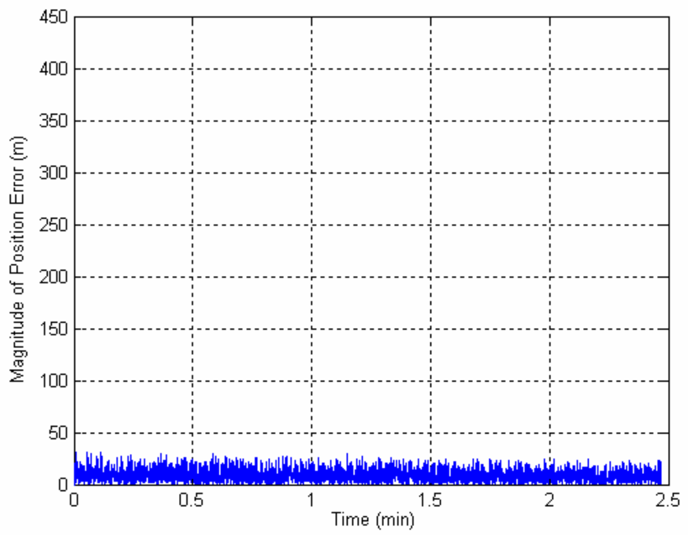


Figure 5–16. Magnitude of Position Error versus Flight Time (IR).

3. Data Fusion

The target track data provided by the RF and IR sensors are fused to obtain a better estimate of the target position. The fusion is accomplished by averaging the track inputs for each coordinate in the Cartesian coordinate system. The resulting magnitude of the fused position error is illustrated in Fig. 5–17. As the figure indicates, the fused target track is better than both RF–only tracks illustrated in Fig. 5–12. For the fused track, the overall error in magnitude of target position remains under approximately 160 m. Furthermore, using different type of sensors is crucial since one type of sensor may provide useful target track data while another type of sensor is under electronic attack or fails.

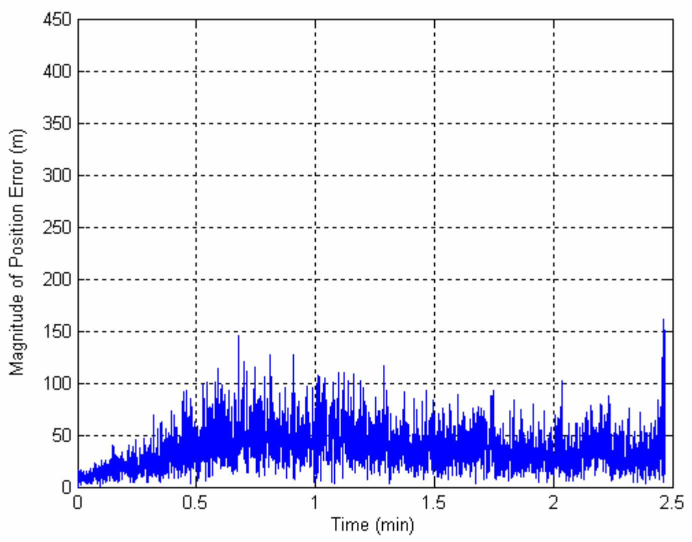


Figure 5–17. Magnitude of Position Error versus Flight Time (Fused).

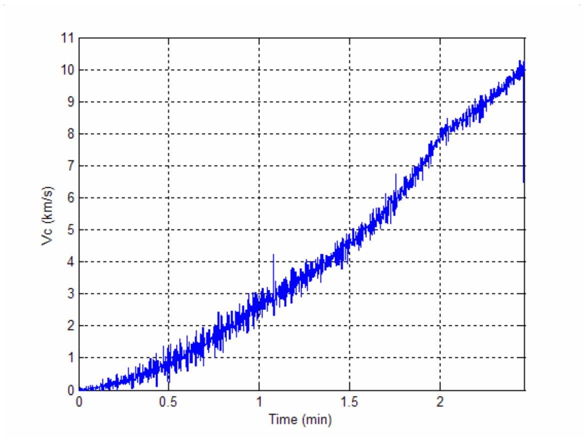
4. Missile Performance

The fused target position is sent to the missile for guidance. Given the noisy target position data, lateral acceleration and lateral divert requirements increase as well as the resulting miss distance. The following test runs were conducted under the conditions listed in Table 5–5.

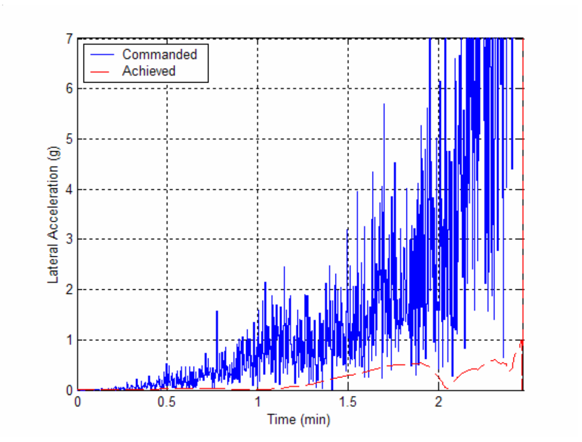
Parameter	**Value**
Data Update Interval	0.15 s
Transmission Delay, τ_t	10 ms
Navigation Coefficient, N'	4
Missile Time Constant, T	5 s

Table 5–5. Missile Test Parameters.

Figure 5–18 shows the closing velocity and lateral acceleration as a function of the flight time. As Fig. 5–18(a) illustrates, the closing velocity is no longer smooth but noisy. With the LOS rate input, this causes a noisy lateral acceleration as shown in Fig. 5–18(b). However, the missile is not able to follow this noisy command input due to the flight system lag. The achieved lateral acceleration is also shown in Fig. 5–18(b). This results in a reasonable achieved control input with a trade–off of miss distance. For this specific run, the intercept time, miss distance, and lateral divert are 2.4692 minutes, 19 m, and 289.8 m/s, respectively.



(a)



(b)

Figure 5–18. Closure and Guidance Characteristics for the Missile Guided by Sensed Target Position Data (a) Closing Velocity versus Flight Time, (b) Lateral Acceleration versus Flight Time.

To further quantify the effect of sensor track quality to miss distance, the simulation was run 1,000 times consecutively with different average track qualities (see Table 5.2). The average magnitude of the target position error and the resulting miss distance was recorded. Figure 5–19 illustrates the simulation and curve fitting results showing the effect of target position error on miss distance. Note the average target position error includes the error due to the transmission delay, which is approximately 40 m. As shown in Fig. 5–19, as the tracking quality decays, the resulting miss distance increases as expected. The simulation results are statistical rather than deterministic since the missile no longer uses the perfect target position data. The resulting data points can be approximated by a quadratic regression curve as shown. The simulation can be used to quantify track quality and its effect on miss distance for any sensor configuration.

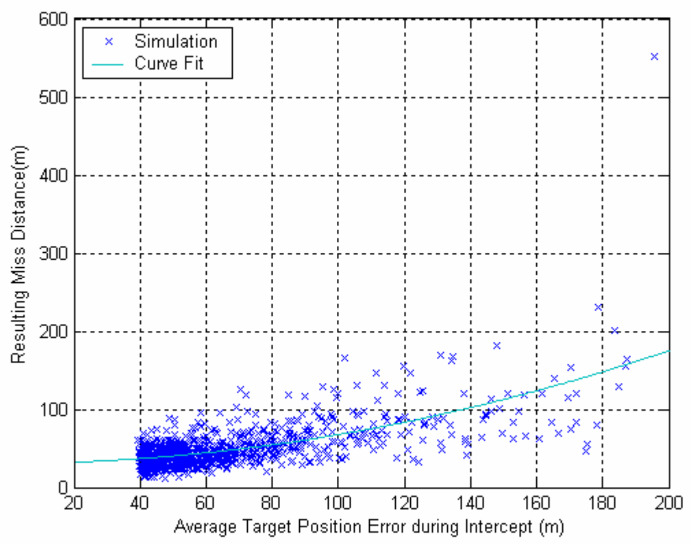


Figure 5–19. Target Position Error versus Miss Distance.

C. SUMMARY

This chapter investigated the issues related to sensors. Different factors, such as transmission delay and tracking inaccuracies, introduce target position errors. The received target position data by the interceptor missile is usually noisy. The missile flight control system filters the noisy inputs while increasing the resulting miss distance. The miss distance increases as the track quality decays. Sensor locations and specifications dictate the quality of the target track. The next step is to investigate the effects of an electronic attack.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. ELECTRONIC ATTACK EFFECTS

This chapter investigates the effects of an electronic attack (EA) during a boost–phase intercept scenario. All sensors including RF and IR can be attacked by the target by releasing RF or IR decoys or radiating jamming signals. All EA is done in order to break the sensor track and/or introduce tracking errors. Consequently, the reduced track accuracy may cause the kill vehicle to be launched at an inappropriate position resulting in a miss.

Most of the countermeasures examined in the literature focus on the mid–course or terminal phase of the attack where techniques may be used easily and effectively by the target. The electronic attack effects are ignored in the boost phase. The objective of this chapter is to investigate the electronic attack effects in the boost phase.

A. EFFECT OF DECOYS

1. Decoy Trajectory

This section examines the use of decoys. Decoy trajectories in the boost phase have major differences with respect to the mid–course or terminal phase counterparts. It has been shown by many papers and reports that submunitions, false targets and decoys can easily overwhelm the defense system [Ref. 6]. This conclusion stems from the fact that any released decoy in the direction of the ballistic missile flight will land at approximately the same targeted point [Ref. 6]. In other words, trajectories of the target itself and the decoys will be the same regardless of the shape or mass of the decoy in the space where aerodynamic effects are not an issue. This fact also led to many creative ideas, such as metallized balloons, shrouds, chaff, and electronic decoys [Ref. 7]. It is greatly emphasized that a ballistic missile can carry and deploy many submunitions or decoys to neutralize the defense.

The first step to investigate the effect of decoys in the boost phase was to predict its trajectory when launched from the target, which was accomplished by modifying the 3D missile simulation by adding a generic decoy model. The decoy trajectory is examined by releasing it at a certain time during the boost phase. The target carries the decoy until it is released so that before the release time it shares the same position and velocity characteristics as the target. After ejection, the decoy begins to decelerate and separate from the target. It should also be emphasized that the motion of the decoy is independent of its mass and physical shape in the exoatmospheric region where most of the intercept takes place. Figure 6–1 illustrates the decoy trajectory being released at $t_{dr} = 90$ s following the target launch.

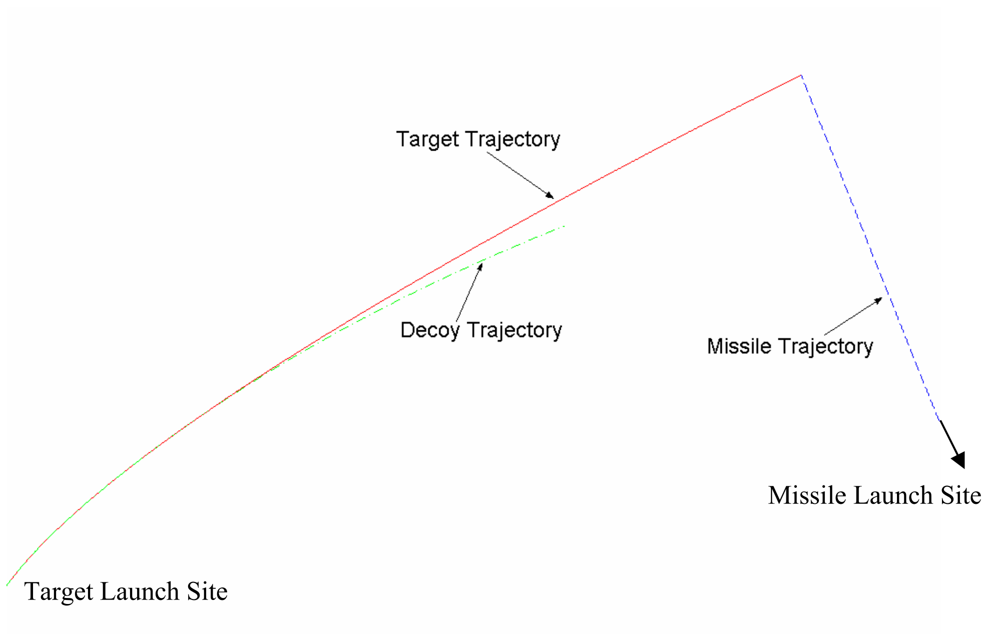


Figure 6–1. Decoy Trajectory (Released at $t = 90$s).

Figure 6–2 shows the height versus ground distance for the target, missile and decoy. As seen in Figs. 6–1 and 6–2, the absence of thrust for the decoy causes separation from the target. This is different from the mid–course case where both target and the decoy move together. Nonetheless, the decoy separation is very smooth at the beginning, which may cause problems for the sensor signal processor.

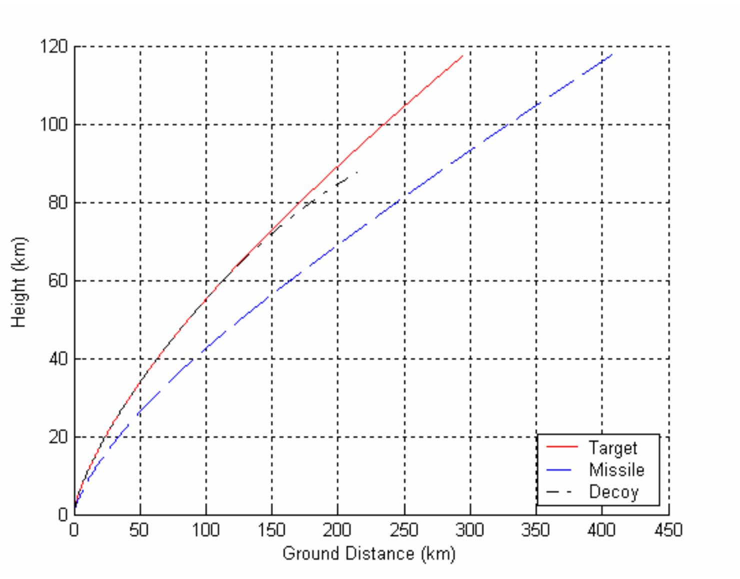


Figure 6–2. Decoy Trajectory (Ground Distance versus Height) for Target, Missile and Decoy.

The target–decoy separation distance is illustrated in Fig. 6–3 as a function of the flight time. As Fig. 6–3 depicts, the target and the decoy move together until release time, which is 90 seconds. After release, the decoy begins to separate due to deceleration.

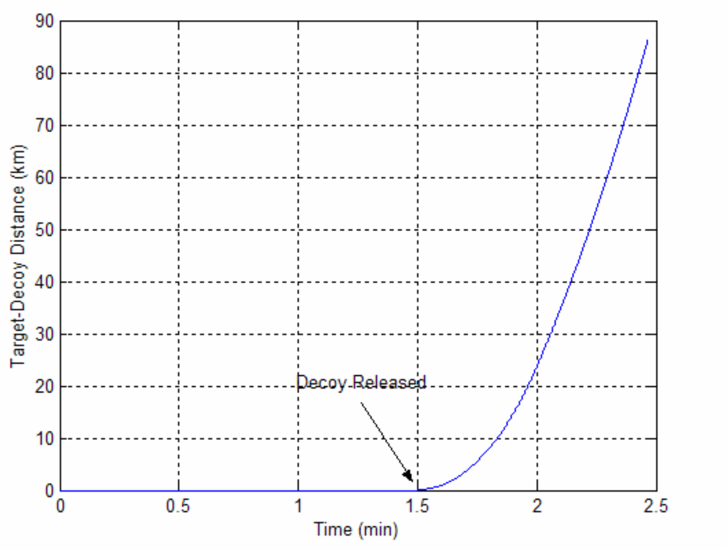


Figure 6–3. Decoy Separation, Target–Decoy Distance versus Flight Time.

Figure 6–4 illustrates the velocity characteristic of the decoy. The decoy begins to decelerate as soon as it separates from the target. This is a great advantage from the defense system point of view. Since the decoy separates from the target, it is then possible to discriminate the decoy and the target from each other in position.

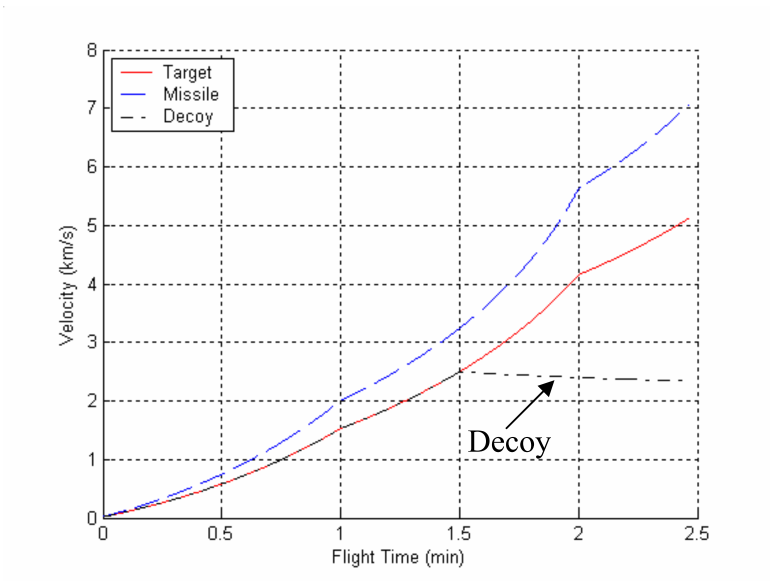


Figure 6–4. Decoy Velocity versus Flight Time.

Decoy trajectories in the boost phase show different characteristics compared to those of the other phases. The most significant difference is that the decoy decelerates and separates from the target quickly. This is a great advantage from the defense system point of view since the difference in physical characteristics of the target and the decoy makes the rejection of decoys possible. However, the initial phase of the decoy separation is smooth, which may introduce temporary track transfer to decoy as discussed in the following sections.

2. IR Decoys (Flare)

IR decoys are employed to steal the track of IR sensors. Pyrotechnic flares are the most typical IR decoys. IR decoys are generally designed to fulfill the following objectives [Ref. 15: p. 291]:

- Must emit sufficient intensity (greater than the platform signature) in order to steal the IR sensor track.
- Must reach the peak intensity as soon as possible and before leaving the sensor field–of–view (FOV).

- Spectral characteristics must resemble that of the target
- Must burn long enough until the target is no longer in FOV.
- Must separate from the platform to a distance, which is no longer inside the effective radius of the warhead.

Most IR decoys burn at 2,000K [Ref. 15: p. 293]. Figure 6–5 compares spectral radiant exitance for the target plume and the decoy assuming that both are blackbodies. The blackbody assumption may not be a proper approach especially for the flare; however, Figure 6–5 is shown not for further analysis but only for a rough spectral comparison. Typical analysis to find the total radiant flux cannot be done since the radiating surface area is not known. Instead, radiation intensity is calculated by using the energy approach. For a pyrotechnic flare, the hypothetical specific intensity in the 3–5 μm band is given as $15.8\,\text{J}/(\text{g}\cdot\text{sr})$. Also, fuel consumption for a typical flare is given as 100 g/s [Ref. 15: p. 301]. By multiplying these two quantities, the radiation intensity for a typical flare in the 3–5 μm band can be calculated as 1.58 kW/sr [Ref. 15: p. 303]. Obviously, this yields almost 33 dB of difference when compared to the 3 MW/sr emitted by the plume in the first stage. This leaves the IR sensor sufficient time to adjust its gain in order to discriminate the decoys from the target. This brief investigation emphasizes the problem with IR decoys being used in the boost phase of a ballistic missile attack. While dealing with mid–course intercepts, cooling, shrouding or IR decoys may introduce bigger problems with the sensors. However, the huge amount of energy emitted from the rocket engines during boost phase makes the usage of IR decoys almost impossible. In other words, by examining the huge energy emitted from the rocket engines, it can be concluded that the boost phase IR detection is very resistant to EA.

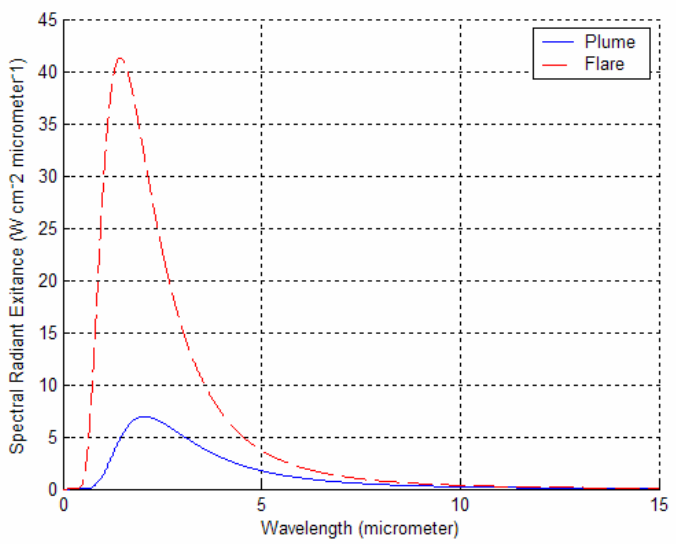


Figure 6–5 Spectral Radiant Exitance of Plume and Flare versus Wavelength.

3. RF Decoys (Chaff)

Passive chaff is generally considered a major EA method. The electronic attack is performed by dispensing a large quantity of conductive dipoles. The radar cross–section of single chaff dipole σ_1 is given by [Ref. 21:p. 242]

$$\sigma_1 = 0.86\lambda^2 \cos^4 \Theta \tag{6-1}$$

where λ is the wavelength (in m) and Θ is the angle between the chaff dipole orientation and the electric field vector. When the chaff dipole is aligned with the electric field vector, (6-1) can be written as [Ref. 21:p. 242]

$$\sigma_1 = 0.86\lambda^2. \tag{6-2}$$

Chaff is dispensed in large quantities. Assuming that the chaff dipoles in the chaff cloud are randomly oriented, the average RCS of the chaff cloud $\bar{\sigma}_j$ has been shown to be [Ref. 21:p. 244]

$$\bar{\sigma}_j = 0.17\lambda^2 N_{je} \tag{6-3}$$

where N_{je} is the number of effective chaff dipoles in the cloud.

Recall that the RCS of the target was predicted as a function of monostatic angle in Chapter IV and was determined as a function of dynamic flight characteristics in Chapter V. According to the findings from Chapter IV and Chapter V, Table 6–1 summarizes the RCS of the target as seen by the RF sensors during the intercept.

	Minimum RCS (m^2)	**Maximum RCS** (m^2)	**Average RCS** (m^2)
RF–1	0.197	71,205	4.479
RF–2	0.336	71,205	5.836

Table 6–1. RCS as Seen by the RF Sensors during Intercept.

The chaff packet dispensed by the target should contain enough dipoles to cover the target in the worst–case scenario where the RF sensors see the target from the specular aspect where the RCS is at its maximum. By using (6-3) and the RCS data, it is possible to calculate the number of chaff dipoles required. For the X–Band radar with an RF frequency of 10 GHz, this corresponds to approximately 465,000,000 chaff dipoles. However, this approach is too conservative. RF sensors see the maximum RCS for very short periods of time during the intercept. By considering the average RCS for the calculation, the number of chaff dipoles is calculated to be 38,000 chaff dipoles.

The number of chaff dipoles in the chaff cloud can be related to the probability that the chaff RCS exceeds the target RCS for the scenario defined in Chapter V. This is accomplished by calculating the probability that the target RCS exceeds a set of values from –10 to +50 dBsm for each time step of the simulation during the intercept. Also, by using (6-3), the number of chaff dipoles required to obtain the RCS values in the same set can be calculated. Figure 6–6 summarizes the results. The black solid line illustrates an average of RF–1 and RF–2. Figure 6–6 shows, for example, that by using a chaff bundle containing 10,000 chaff dipoles, the cloud is able to cover the target RCS with a probability of 0.6. Similarly, by using a 1,000,000–dipole bundle protects the target 0.8 of the time during the intercept. Figure 6–6 can also be used backwards. For example, given that the target RCS should be covered by a chaff cloud, with a probability of 0.9, it requires 1,862,087 chaff dipoles.

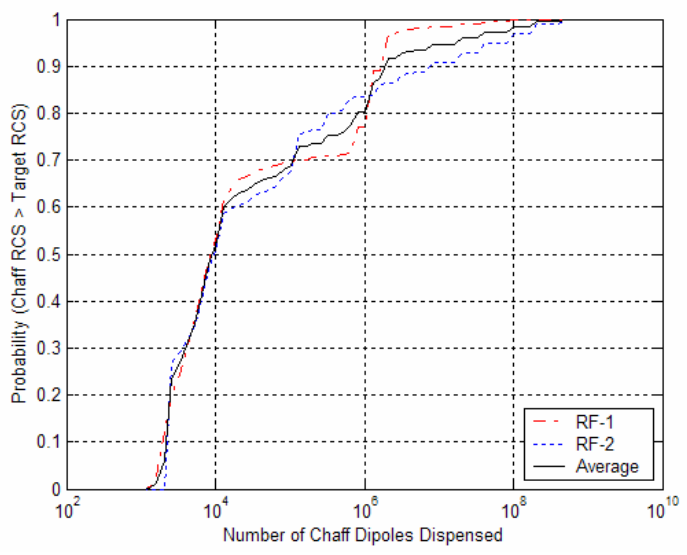


Figure 6–6. Probability of Average RCS of Chaff Cloud Exceeds the Target RCS versus Number of Chaff Dipoles Dispensed.

Recall from Chapter V that the RF sensors were located at approximately optimal positions where they can see the target from a good aspect, thus maximizing the RCS. Knowing the maximum possible RCS, a theoretical limit on the number of chaff dipoles can be calculated as 518,800,000.

In order to investigate the feasibility that the target dispenses a certain number of chaff dipoles, the consideration was that a single chaff dipole had a length of 1.5 cm and a thickness of 1 mm. The rectangular volume occupied by a single chaff element equals $15\times10^{-9}\ \text{m}^3$. Assuming a probability of success of 0.9, 1,862,087 dipoles must fit into a volume of $0.028\,\text{m}^3$. This corresponds to a cube that has approximate side lengths of 30 cm. The calculation shows that, even for a high probability of success, the price paid is low. This means that, carrying and dispensing chaff for ICBMs is feasible and probable. This increases the probability that a considerable percentage of electronic attack efforts will target RF sensors.

4. Track Transfer to Decoy

As examined in Section A–2, stealing an IR track is not very likely due to the enormous amount of energy emitted by the target plume. This scenario assumes that the decoy captures only the RF sensor tracks. The decoy is released 90 seconds after the target launch. Figure 6–7 illustrates this scenario in 3D. As seen in Fig. 6–7, the guidance law adapts to the new situation immediately. Since the track data is fused by simply averaging the sensor inputs, the resulting target position occurs at an average point. This type of scenario results in the failure to intercept the target. In this case, the miss distance is measured as 67 km.

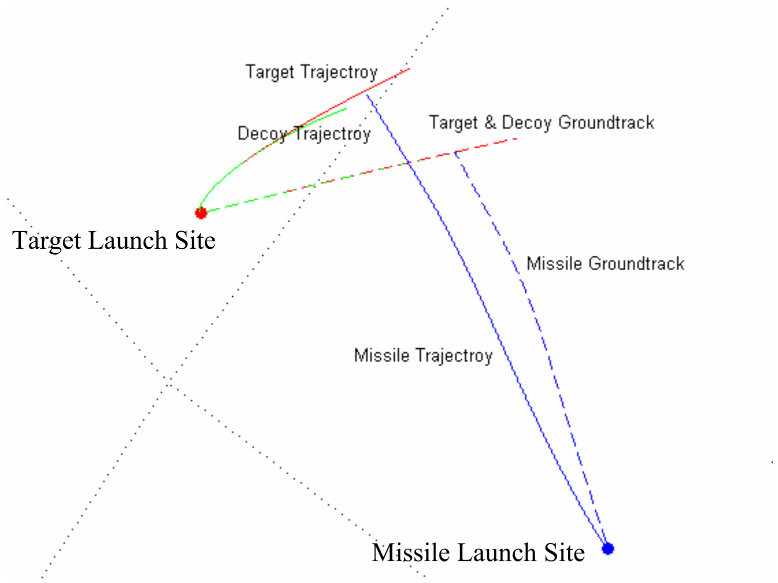


Figure 6–7. 3D Overview of the Intercept: Both RF Sensor Tracks Captured by Decoy.

Since the time–to–go decreases with respect to the initial heading error for the new geometry, a high price is paid in terms of lateral acceleration requirements. Figure 6–8 illustrates the lateral acceleration as a function of the flight time. The track transfer to decoy results in excessive lateral acceleration commands. This is not important when the track remained on the decoy since the intercept already fails. However, assuming that the target may be reacquired, excessive lateral acceleration commands cause a loss of available energy.

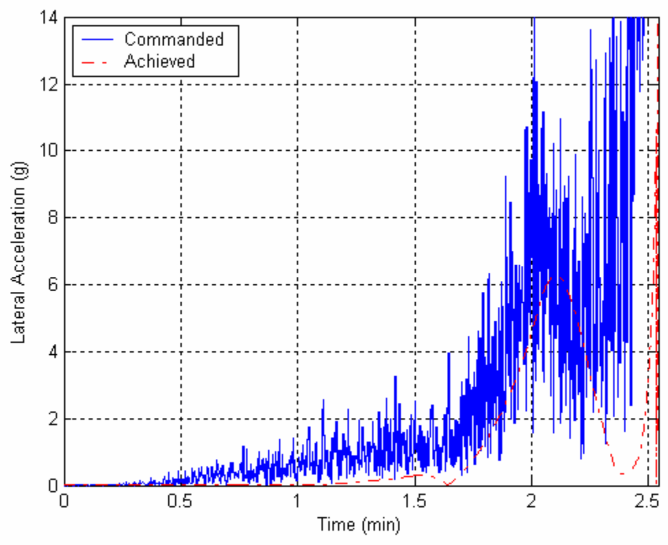


Figure 6–8 Increase in the Lateral Acceleration Requirements when both RF Sensor Tracks Captured by the Decoy.

Another possible scenario is when either RF–1 track or RF–2 track is captured by the decoy. Figure 6–9 depicts the situation where either RF–1 or RF–2 track transfers to decoy. In this case, only one target position component of the fused data is in error. The resulting miss distance is 32 km, which is too large for the proper launch of a kill vehicle.

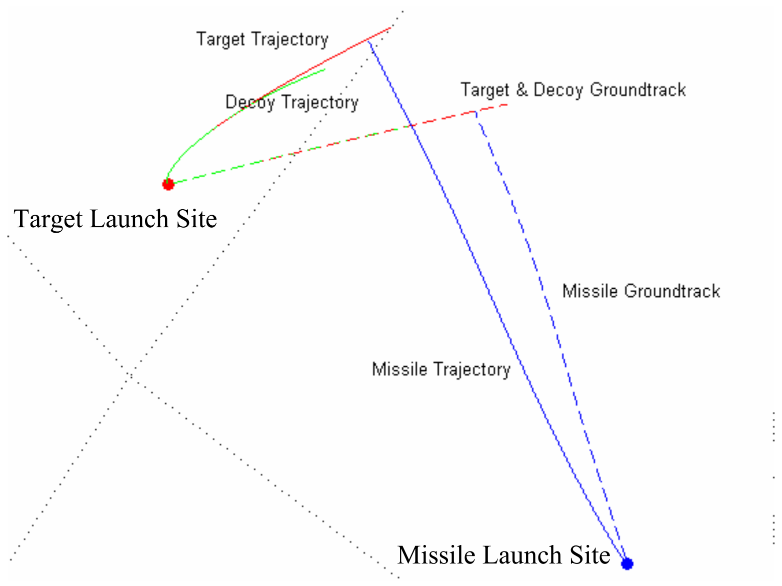


Figure 6–9. 3D Overview of the Intercept: Either RF–1 or RF–2 Track Captured by Decoy.

Lateral acceleration requirements are improved compared to the case where both RF sensor tracks are captured. For example, around 2 minutes where the achieved acceleration reaches a peak, the lateral acceleration is decreased from 6 g to 4 g as shown in Fig. 6–10. However, this is still higher than the case in which no decoys exist.

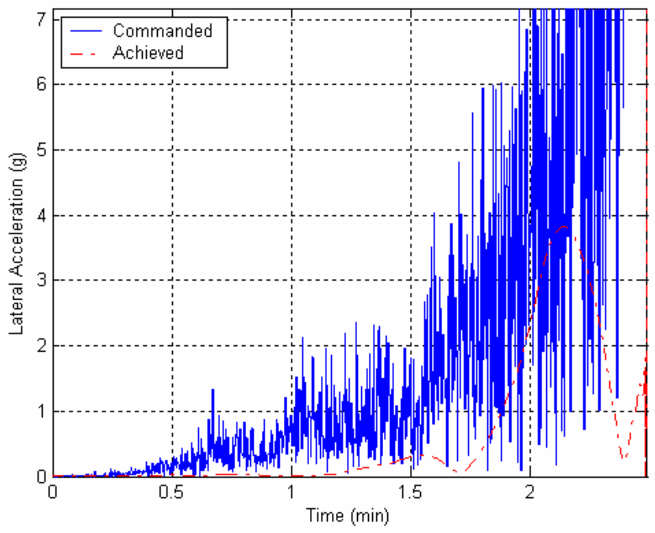
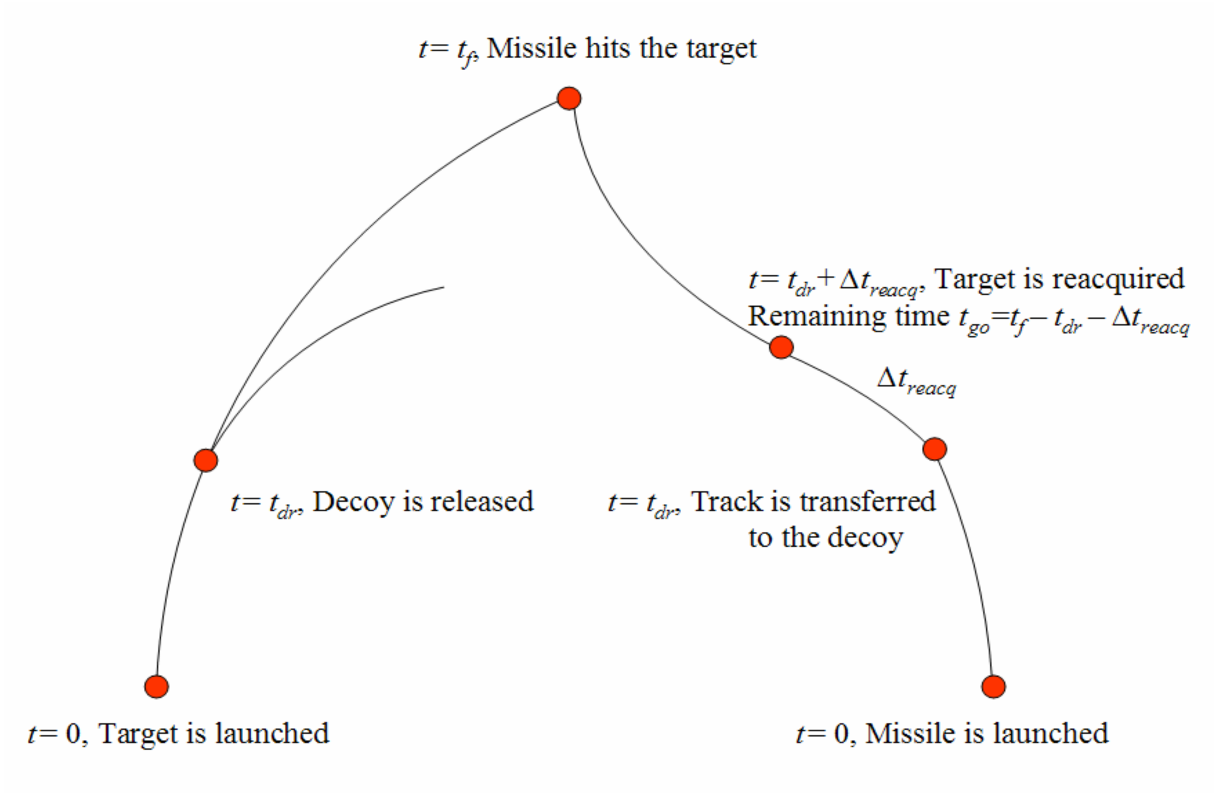


Figure 6–10. Lateral Acceleration versus Flight Time, Either RF–1 or RF–2 Track Captured by Decoy.

Sensed target characteristics, such as target acceleration, differ when the track is transferred to the decoy. It is very probable that the sensors are equipped with good signal processing capability by which they can discriminate the target from the decoy. In fact, this is a normal requirement since, when jettisoned stages are present, they are treated as target maneuvers and their trajectories are similar to those of decoys.

The next objective is to examine where the decoy captures the sensor track temporarily, but the radar signal processor has an electronic protection routine and reacquires the target. Figure 6–11 depicts a scenario where the sensors reacquire the target. In this scenario, missile and target are launched at $t = 0$. When decoy release time t_{dr} is reached, the decoy is released causing an immediate track transfer to the decoy. The signal processors of the RF sensors process the data and reacquire the target in Δt_{reacq} seconds. From this point, missile resumes to follow the target.



$t = t_f$, Missile hits the target
$t = t_{dr} + \Delta t_{reacq}$, Target is reacquired
Remaining time $t_{go} = t_f - t_{dr} - \Delta t_{reacq}$
Δt_{reacq}
$t = t_{dr}$, Decoy is released
$t = t_{dr}$, Track is transferred to the decoy
$t = 0$, Target is launched
$t = 0$, Missile is launched

Figure 6–11. Scenario for Track Transfer to Decoy and Consecutive Reacquisition of the Target.

Determination of miss distance in the presence of reacquisition is a complicated problem. Revisiting Fig. 3–12 illustrates the effect of system dynamics on the miss distance. As Fig. 3–12 depicts, if the missile time constant T is less than 10% of the flight time t_f, (or in other words, time–to–go t_{go} is more than 10 times the missile time constant), the miss distance is negligible ($t_f = t_{go}$ where no decoy exists). In the test case, the missile flight time is approximately $t_f = 150$ s and a time constant is $T = 5$ s. The missile with the time constant $T = 5$ s requires at least 50 seconds of time–to–go t_{go} in order to keep the miss distance small. This means that if decoy release time plus reacquisition time $t_{dr} + \Delta t_{reacq}$ is less than 100 seconds, this will not affect the miss distance significantly. This demonstrates that the target should be reacquired at no later than 100 seconds during the intercept, or otherwise the miss distance will be significant. Figure 6–12 illustrates the miss distance as a function of both decoy release time and reacquisition time. The figure is constructed by setting decoy release times between 90 and 150 seconds and reacquisition times between 5 and 20 seconds in 1–second intervals. For each case, two samples are collected in order to reduce the error magnitude introduced by the sensor inaccuracies.

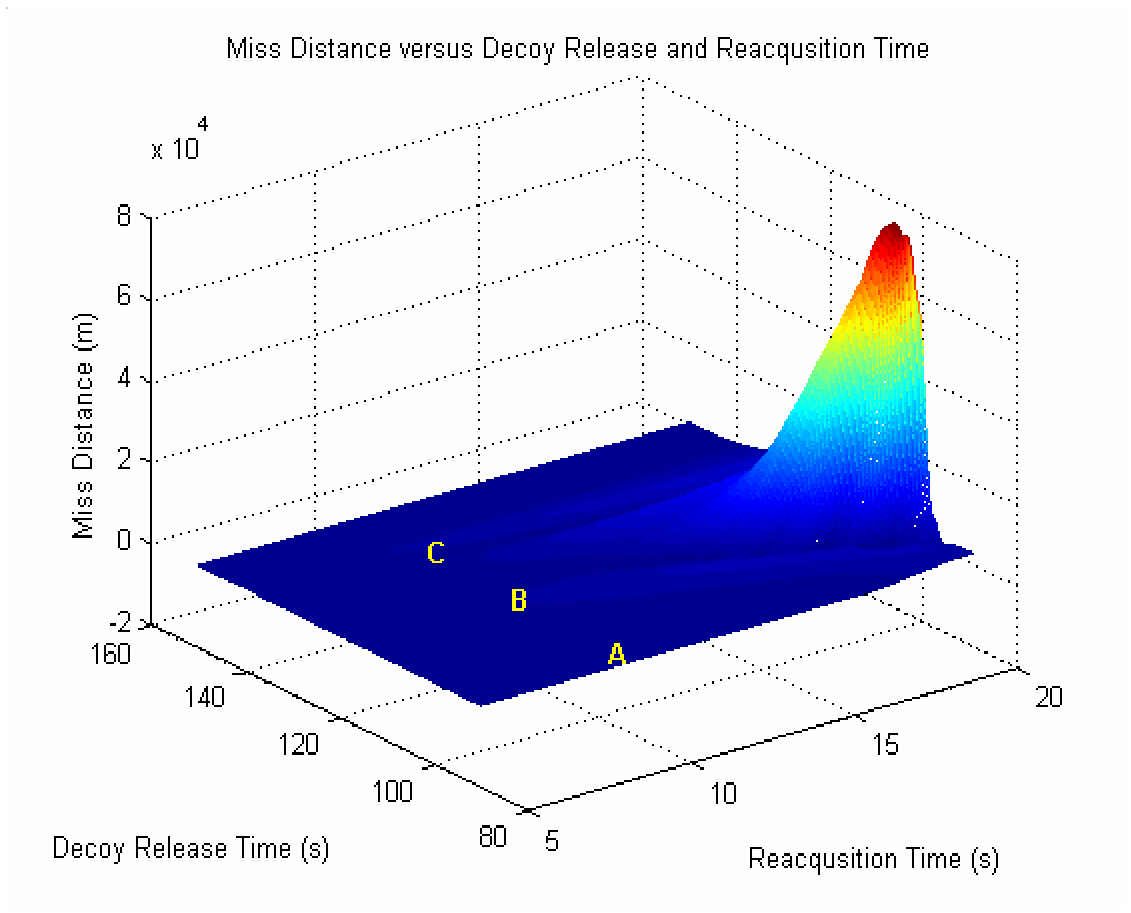


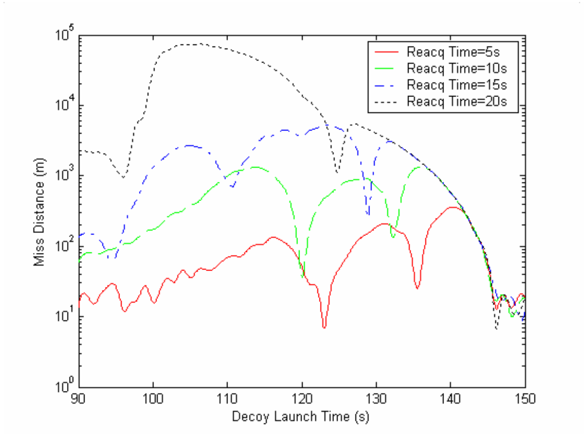
Figure 6–12. Miss Distance as a Function of Decoy Release and Reacquisition Time.

As Fig. 6–12 illustrates, in region A where the missile has enough time to adapt to the new situation after reacquisition (time–to–go t_{go} is greater than 50 s), the miss distance is only a function of the usual errors introduced by the sensors. In this region, unless the decoy release time plus reacquisition time $t_{dr} + \Delta t_{reacq}$ exceeds 10 times missile time constant T, the resulting miss distance is independent of the decoy effect.

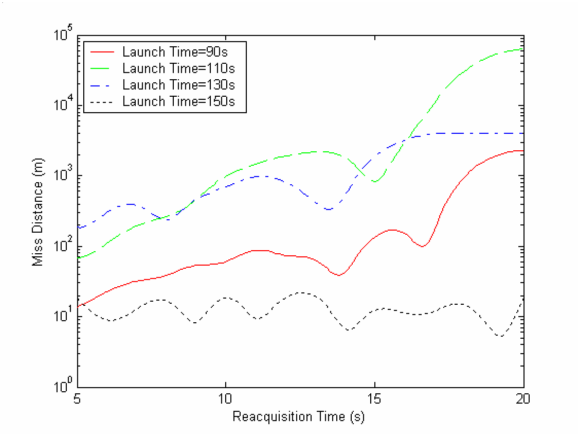
In region B, the time–to–go time constant ratio drops below 50, and the miss distance becomes affected by the reacquisition time. Given the same decoy release time, the miss distance values generally follow the characteristics of miss distance versus time, which was shown in Fig. 3–12. In this region, as the reacquisition time increases, the miss distance becomes unacceptable. If the sensors cannot reacquire until the missile detonates, the miss distance may be on the order of 70 km.

In region C, the decoy is launched too late; so that the miss distance is small.

Using two–dimensional projections of Fig. 6–12 may help explain the findings better. Figure 6–13 shows the miss distance with respect to the decoy launch time and reacquisition time. The miss distance is measured as a function of decoy launch time and the different curves depict different reacquisition times. As decoy launch time t_{dr} increases, the miss distance increases until the intercept time when it is too late to launch the decoy. The general characteristic of the curve confirms the results from Fig. 3–12 where the curve peaks as a function of the ratio of missile time constant to the time–to–go. As the reacquisition time Δt_{reacq} increases, the curves shift since the sum of the decoy release time and reacquisition time $t_{dr} + \Delta t_{reacq}$ changes. At 20 seconds of reacquisition time Δt_{reacq}, the curve is usually a function of the decoy–target distance since the sensors can never reacquire the target before detonation. This may increase the miss distance up to 70 km. Figure 6–13 illustrates the miss distance as a function of reacquisition time Δt_{reacq} for different decoy launch times t_{dr}. When the decoy launch time t_{dr} is 150 seconds, the curve is only a function of the usual errors introduced by sensors since it is too late to launch the decoy. The other curves, however, depict that as the reacquisition time Δt_{reacq} increases, the miss distance increases following the missile time constant to time–to–go ratio.



(a)



(b)

Figure 6–13 Miss Distance as a Function of (a) Decoy Release Time, (b) Reacquisition Time.

This brief investigation reveals the following important results:

- Since the decoy separates from the target smoothly, there is always a chance of transferring the sensor lock to the decoy.
- If the sensor is not able to reacquire the target, the miss distance may be in tens of kilometers depending on when the decoy is launched and which sensors are affected.
- Sensors may be able to reacquire the target. Even if the target is reacquired, there is always a chance of failure depending on when the decoy is launched and how long it takes for the sensors to reacquire.
- In the case where the sensors can reacquire the target, the miss distance is a function of time–to–go and missile time constant.

B. EFFECT OF NOISE JAMMING

This section examines the effect of target noise jamming on the intercept. The target is able to radiate noise jamming in the RF sensor band (X–Band) in an attempt to accomplish self–screening. In the case, where jamming is not present, the signal power returned from the target competes with the noise at the receiver input. However, if effective noise jamming is present, the useful signal will be overpowered by the jamming signal. The jamming power at the radar antenna $(P_j)_{in}$ is given by [Ref. 21: p. 170]

$$(P_j)_{in} = \frac{P_j G_j}{4\pi D_j^2} A_s F_s^2(\Phi_j, \Theta_j) F_j^2(\Phi_s, \Theta_s) \gamma_j \frac{\Delta f_{rec}}{\Delta f_j} \Gamma_{JS,radar}^2 10^{-0.1\alpha L_j} \qquad (6\text{-}4)$$

where P_j is the jammer peak power (in W); G_j is the jammer antenna gain; D_j is the jammer–to–victim radar distance (in m); F_s and F_j are the normalized antenna patterns of the radar and jammer with respect to each other; Φ_j, Φ_s and θ_j, θ_s are the azimuth and elevation angles, respectively; γ_j is the polarization coefficient; Δf_{rec} and Δf_j are the radar and jammer bandwidths (in Hz), respectively; Γ is the propagation factor; α is the attenuation coefficient (in 1/m); L_j is the path length (in m); and A_s is the radar antenna effective aperture given by [Ref. 21: p. 169]

$$A_s = \frac{G_s \lambda^2}{4\pi} \tag{6-5}$$

where G_s is the radar antenna gain.

The following assumptions are made to simplify the problem:

- $F_s = F_j = 1$ (assumes that the jammer antenna is pointing at the radar and the radar antenna is pointing at the jammer).
- $\gamma_j = 1$ (assumes that the jammer signal polarization is aligned with the radar signal polarization).
- $\Gamma = 1$ (multipath effects are negligible due to the look–up nature of the scenario).
- $\alpha = 0$ (atmospheric attenuation is negligible with respect to the free space loss due to the frequency of the emitter)

Following the assumptions above, (6-4) can be written as

$$(P_j)_{in} = \frac{P_j G_j G_s \lambda^2 \Delta f_{rec}}{(4\pi)^2 D_j^2 \Delta f_j}. \tag{6-6}$$

Similarly, the power at the radar receiver input $(P_s)_{in}$ reflected back from the target is [Ref. 21:p. 171]

$$(P_s)_{in} = \frac{P_s G_s^2 \sigma \lambda^2}{(4\pi)^3 D_s^4} \tag{6-7}$$

where D_s is the target–to–radar distance and σ is the target radar cross section.

Neglecting the receiver noise power, which is small with respect to the jamming signal received by the radar, the signal–to–jam ratio S/J used to calculate the tracking accuracy of the sensors can be expressed as

$$S/J = \left(\frac{P_s}{P_j}\right)_{in} = \frac{P_s G_s}{P_j G_j} \frac{\sigma}{4\pi} \frac{D_j^2}{D_s^4} \frac{\Delta f_j}{\Delta f_{rec}}. \tag{6-8}$$

It is possible to make further simplifications by using the following assumptions:

- $D = D_S = D_j$ (jammer is co–located with the target).
- $G_j = 1$ (jammer is an isotropic radiator).

In this case, (6-8) simplifies to

$$S/J = \frac{P_s G_s \sigma \Delta f_j}{4\pi P_j D^2 \Delta f_{rec}} \tag{6-9}$$

where D is the sensor–to–target distance.

Through simulation, it is possible to quantify the S/J ratio during the intercept for both RF sensors. It is a good assumption that the jammer design will consider a large bandwidth since the actual RF sensor specifications are not known. This is an advantage from the interceptor's point of view. The generic jammer uses the bandwidths shown in Table 6–2. The jammer should consider enough bandwidth and power density to mask the target RCS. Assuming that the jammer designer decided to jam the X–Band radar, the specific frequency is not known. In addition, there is more than one RF sensor, which may be operating at different frequencies. The simulation uses a jammer, which covers the X–Band completely.

	Frequency Range	**Bandwidth**
X–Band Only	8–12 GHz	4 GHz
X and C Band	4–12 GHz	8 GHz
X, C and S Band	2–12 GHz	10 GHz
X, C, S and L Band	1–12 GHz	11 GHz

Table 6–2. Possible Bandwidths to be Considered by the Jammer.

Figure 6–14 illustrates the S/J ratio when a 1–kW jammer is used. The jammer causes the S/J to decay with respect to the case where no jamming is present which was illustrated in Fig. 5–11.

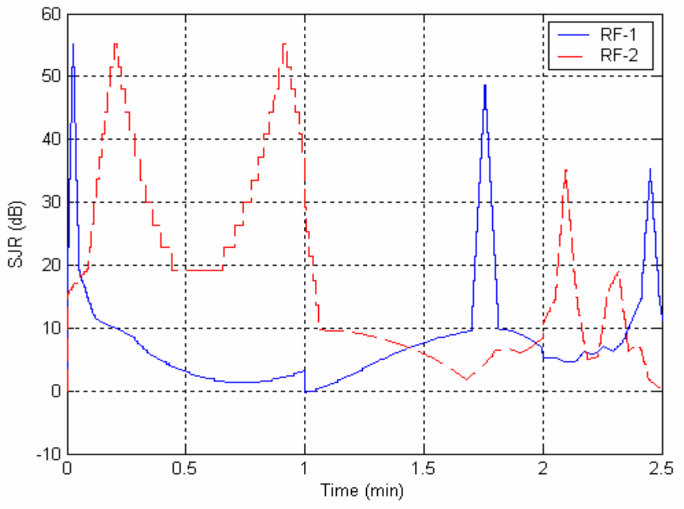
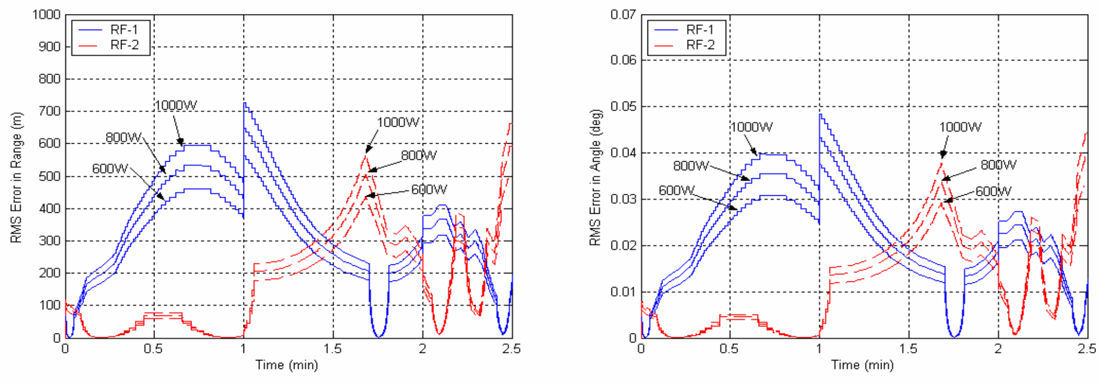


Figure 6–14. S/J Ratio during the Intercept for 1kW Jammer.

Two factors affecting the signal–to–jam ratio are the jammer power and the jammer bandwidth. To investigate these effects, the simulation was used to quantify the tracking errors introduced. As Fig. 6–15 depicts, an increase in jammer power reduces the target track quality by increasing the RMS error in range (Fig. 6–15(a)) and angle (Fig. 6–15(b)).



(a) (b)

Figure 6–15. Effect of Jammer Power (4 GHz Bandwidth): RMS Error versus Flight Time in, (a) Range, (b) Angle.

Figure 6–16 depicts that as the jammer bandwidth decreases, the tracking quality decays since the jammer is better focused on the emitter's bandwidth.

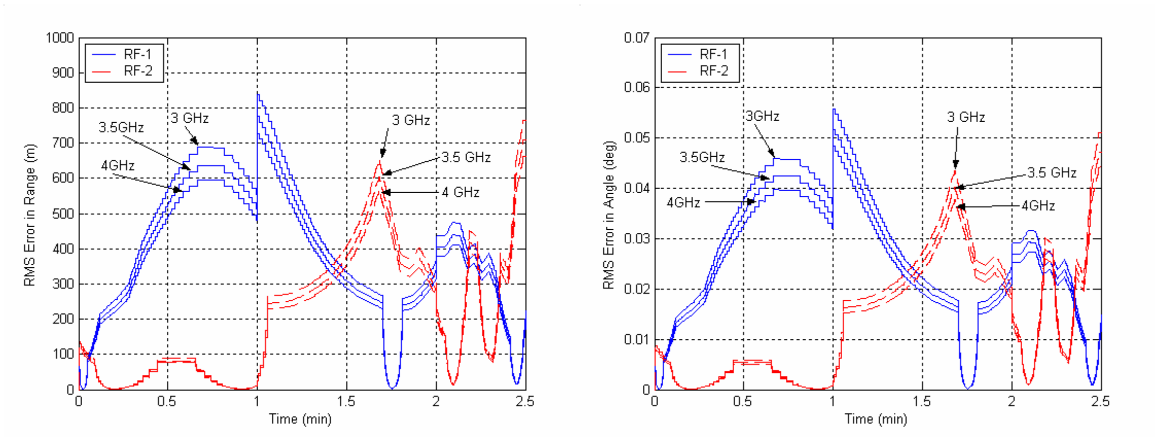


Figure 6–16. Effect of Jammer Bandwidth (1 kW Power): RMS Error versus Flight Time in, (a) Range, (b) Angle.

It is possible to quantify the miss distance caused by the jammer characteristics using the simulation. Figure 6–17 illustrates the effect of jamming power density (in μW/Hz) on the miss distance. Figure 6–17 illustrates the data points coming from the simulation and the quadratic regression line fitted to the data. It can be concluded that the noise jamming may significantly affect the RF sensor accuracy and miss distance. Increasing the jamming power density beyond that shown in Fig. 6–17 will cause the RF sensor to switch to a home–on–jam mode and only an angle track will be possible. In this case, two RF sensors will have to use triangulation to derive the target's appropriate position.

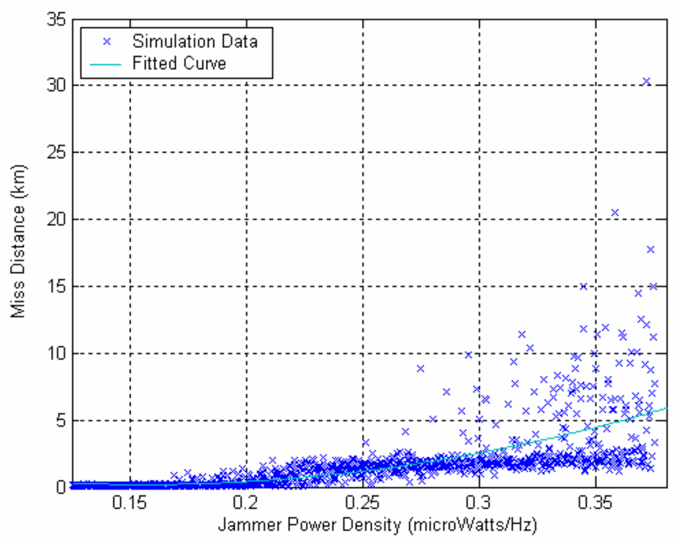


Figure 6–17. Effect of Jamming on Miss Distance.

C. SUMMARY

This chapter investigated the electronic attack effects in the boost phase. The examination included the effects of the IR decoys on the IR sensors and RF decoys on the RF sensors as well as noise jamming effects. This concludes the investigation; however, it has not been possible to include all electronic attack types and their effects in the boost phase.

VII. CONCLUSIONS

A. SUMMARY OF THE WORK

In this research, investigation into the many aspects of ballistic missile boost phase intercept problem was conducted. A mathematical model, which addresses many important elements in three–dimensional space, was developed in MATLAB to achieve the results.

The work developed a multi–stage boosting target model. The simulation was run for a complete three–phase intercontinental attack. The results provided valuable findings for the understanding of the boost, midcourse and terminal phases of the attack.

The next step in the investigation was modeling the physical characteristics of the intercepting missile. To accomplish this, we developed a multi–stage, boosting missile capable of intercepting the previously constructed ballistic target model. The missile requirements were investigated in terms of capability and position. The study also investigated the effects of distance and angular deviations as well as the launch delay.

The monostatic radar cross–section (RCS) and plume characteristics of the target were investigated. After predicting the RCS with the POFACETS software, the plume characteristics were investigated by assuming the plume to be a simple isotropic radiator with a given surface area. Planck's equation calculated the radiation intensity within a given IR band of interest.

The investigation was continued by defining the sensors. We used mathematical models to quantify the tracking errors introduced by the sensors. This part of the investigation examined the tracking issues that led to the miss distance.

Finally, the target's use of electronic attack during the boost phase was investigated.

B. SIGNIFICANT RESULTS

An intercontinental ballistic missile should reach a velocity on the order of 6 to 7 km/s at burnout depending on the distance of its target from the launch site. After burnout, the target starts decelerating at an approximate height of 250 km until it reaches an apogee of approximately 1,600 km. Later, it starts descending and accelerating due to gravity.

The boost phase where the target accelerates lasts for a relatively short time with respect to the overall flight time of the ballistic missile. Having the objective of intercepting the target during the boost phase introduces significant detection and decision time limitations on the defense system.

Excessive acceleration levels as well as discontinuities make the boost phase intercept problem more difficult compared to the mid–course or terminal phase intercept scenarios. The target acceleration perpendicular to the line–of–sight (LOS), which is also known as a target maneuver, can be up to 5 g. The acceleration profile also has discontinuities at stage changes.

For the realistic target model, proportional navigation did not achieve a zero effort miss; however, lateral acceleration and lateral divert requirements were reasonable. This investigation does not cover the terminal phase of the intercept, which is performed by the kill–vehicle. Except for the terminal phase saturation, proportional navigation worked well against the realistic target. Proportional navigation can be a good option while guiding the interceptor until the kill vehicle launch.

The launch location in angle and distance from the target launch site and the capability of the missile and launch delay are significant factors affecting a successful intercept. The capability of the missile became very important when position and launch delay deviations were introduced. Generally, the more capable the missile, the more tolerable it is to less than ideal circumstances.

Positional advantage was the best when the interceptor missile was located directly in the attack direction and with zero launch delay. As the deviations from the ideal conditions were introduced, location and launch delay tolerances decayed quickly. It was shown that, given an angular deviation and/or acceptable launch delay, the maximum distance at which the missile can be located could be estimated by using the model.

The target aspect angle seen by the RF sensor significantly affects the RCS value. RCS can be improved by looking at the target from a specular direction: the side or the bottom. In the RF sensor design, the power requirements must take into account a backscatter performance on the order of –10 to –20 dBsm. The optimum RF sensor bearing for maximizing target backscatter was calculated as 75° measured from the direction of the attack. RF sensor accuracy is a function of the SNR. SNR is affected by radar design specifications as well as the physical parameters, such as RCS and range to target.

The plume is the primary source of radiation for the IR sensors. The missile plume emits a large amount of energy in the IR band. This research showed that the radiation intensity might be up to 3 MW/sr in the first stage of the missile. IR sensors were considered as step–stare focal plane arrays (FPA), which were mounted on LEO satellites. IR sensor accuracy depends on the IFOV of the sensors as well as the sensor height.

Decoy trajectories in the boost phase have major differences with respect to the mid–course or terminal phase counterparts. In the boost phase, the decoy begins to decelerate and separate from the real target following ejection. The decoy separation and deceleration are a big advantage from the defense system point of view. Since the decoy separates from the target, it may be possible to discriminate the decoy and the real target from each other in position. Since the decoy separates from the target smoothly, there is always a chance of transferring the sensor lock to the decoy. If the sensor is not able to reacquire the real target, the miss distance may be on the order of tens of kilometers depending on when the decoy is launched and which sensors are affected. Sensors may be able to reacquire the target. Even if the target is reacquired, there is always a chance of failure depending on when the decoy is launched and how long it takes for the sensors to reacquire the target.

Use of IR decoys is extremely ineffective due to the large amount of energy radiated by the missile plume upon launch. Designing LEO sensors with narrow instantaneous fields of views may increase the signal–to–clutter ratio greatly during intercept. IR sensors, being more resistant to the electronic attack, are the key sensors to keep the miss distance at acceptable levels.

Chaff packages having enough dipoles to cover the target return can be carried and dispensed during the boost phase. This may be very effective in terms of covering the target skin return. The infeasibility of using IR decoys due to huge plume emission from the target increases the probability that a considerable percentage of electronic attack efforts will target RF sensors.

Active jamming may be extremely effective.

C. SUGGESTIONS FOR FUTURE WORK

Data fusion and kill vehicle flight modeling were not investigated in depth in this thesis. In a related study, Humali [Ref. 22] examined several issues related to RF and IR sensor data fusion. Bardanis [Ref. 23] explored some issues related to kill vehicle modeling and guidance. A future effort may extend work supported in this thesis and by [Ref. 22] and [Ref. 23] to study the kill vehicle modeling in detail and investigate the effectiveness of different fusion algorithms to accomplish a successful target hit.

An application of the Kalman filter for the missile model as well as electronic attack may be considered for future work. Since the control inputs are filtered only by the missile flight control system, existing implementation of the model is somewhat limited. A Kalman filter may be a more effective tool. Furthermore, the effect of the Kalman filter on the control system performance in the case of an electronic attack would be a very interesting topic as part of this future study.

This thesis devoted most of its effort to the development of target, missile and sensor models, because without an accurate model, it is not possible to simulate many aspects of the boost phase ballistic missile intercept. As a result, the investigation into the electronic attack mechanisms reported in this thesis is rather preliminary. A future thesis project might extend this effort to obtain a more in–depth analysis of electronic attack effects on the boost phase.

Although the IR sensors are assumed to be fixed over the target and the missile launch sites for simplicity, it is possible to move the IR sensors in orbit by modifying related parameters in the simulation. The simulation code from this thesis supports moving sensors in the orbit. This may help calculate the number of satellites required in the orbit, which is an important subject in the investigation of requirements of boost phase ballistic missile intercept systems.

This research focused on a single target launch. Future work may consider multiple target launches. This can be done by modifying the existing simulation code.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A CODE FLOWCHART

This appendix includes the flowchart for the MATLAB code. The flowchart can be used in conjunction with the code listed in Appendix B to understand and modify the code for future research.

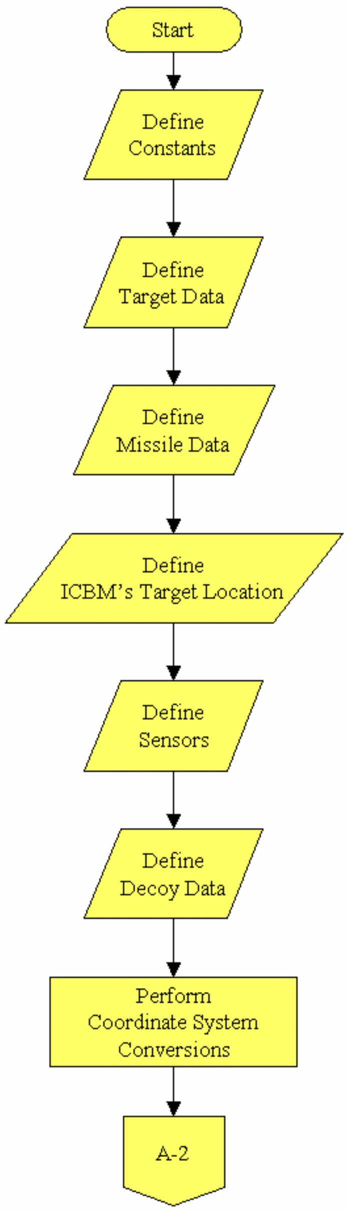


Figure A–1. Code Flowchart (1 of 7).

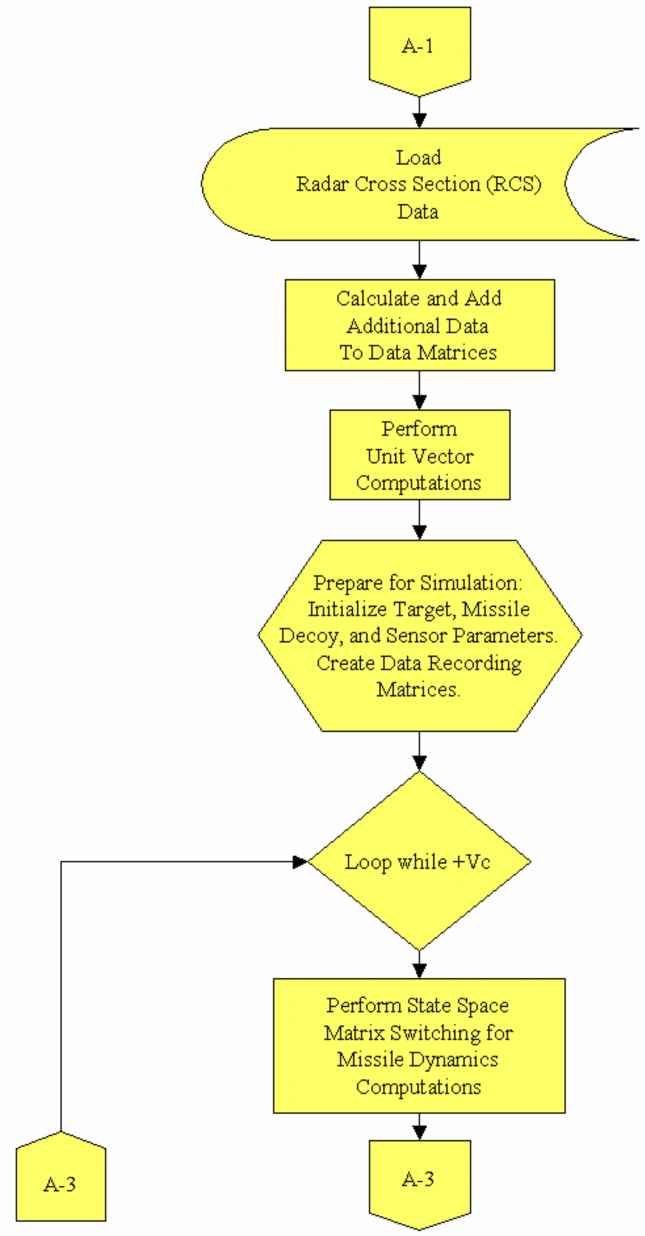


Figure A–2. Code Flowchart (2 of 7).

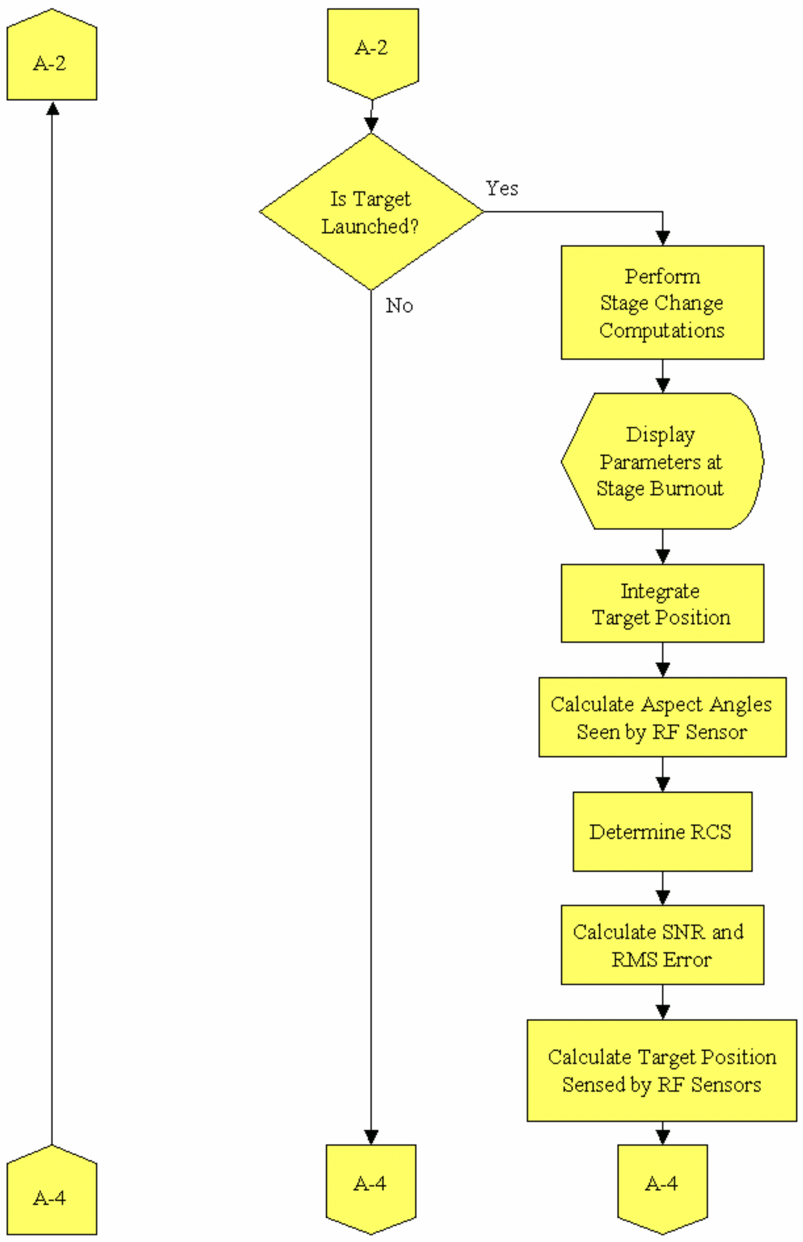

A-2
A-2
Is Target Launched?
Yes
No
Perform Stage Change Computations
Display Parameters at Stage Burnout
Integrate Target Position
Calculate Aspect Angles Seen by RF Sensor
Determine RCS
Calculate SNR and RMS Error
Calculate Target Position Sensed by RF Sensors
A-4
A-4
A-4

Figure A–3. Code Flowchart (3 of 7).

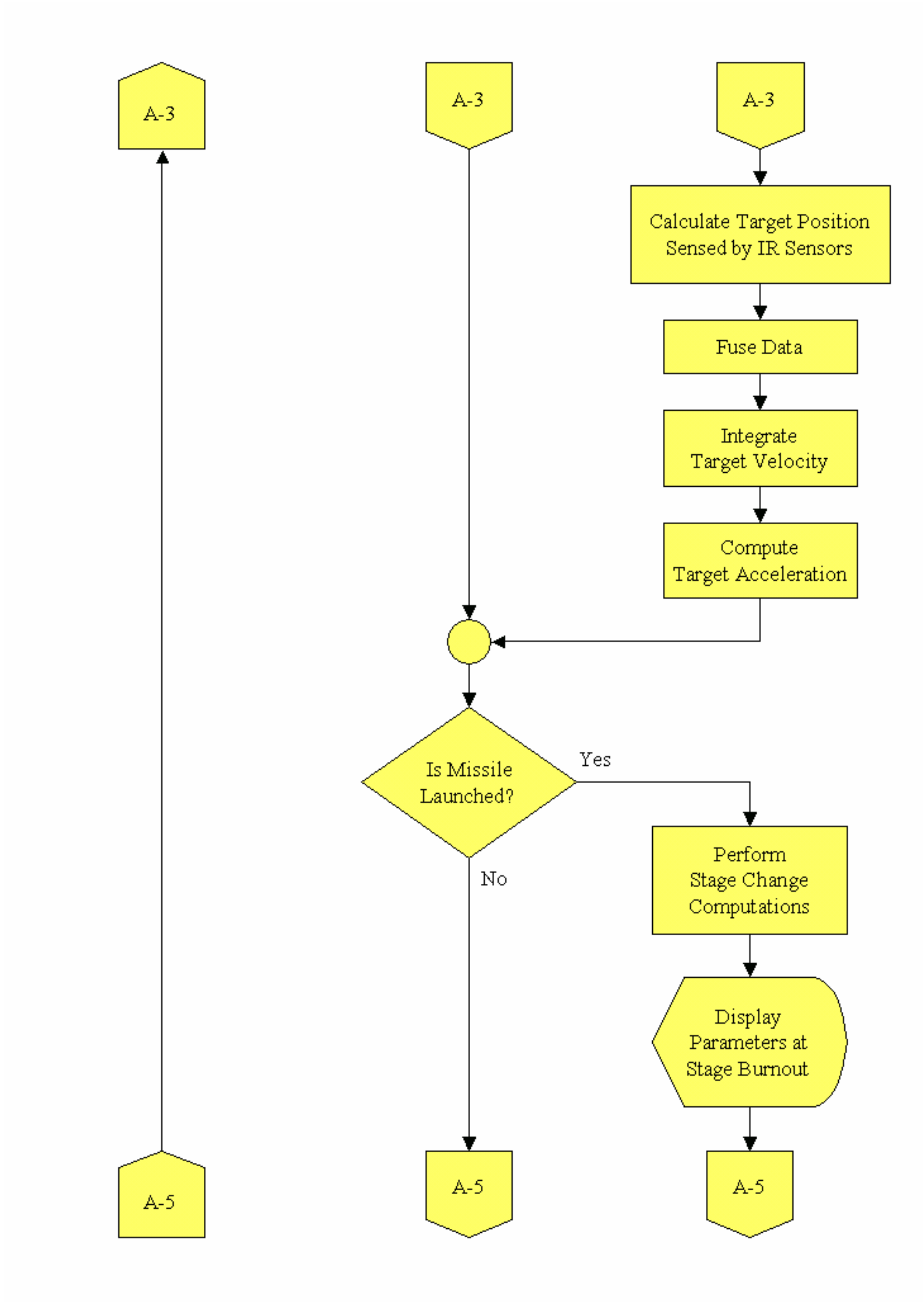

A-3
A-3
A-3
Calculate Target Position Sensed by IR Sensors
Fuse Data
Integrate Target Velocity
Compute Target Acceleration
Is Missile Launched?
Yes
No
Perform Stage Change Computations
Display Parameters at Stage Burnout
A-5
A-5
A-5

Figure A–4. Code Flowchart (4 of 7).

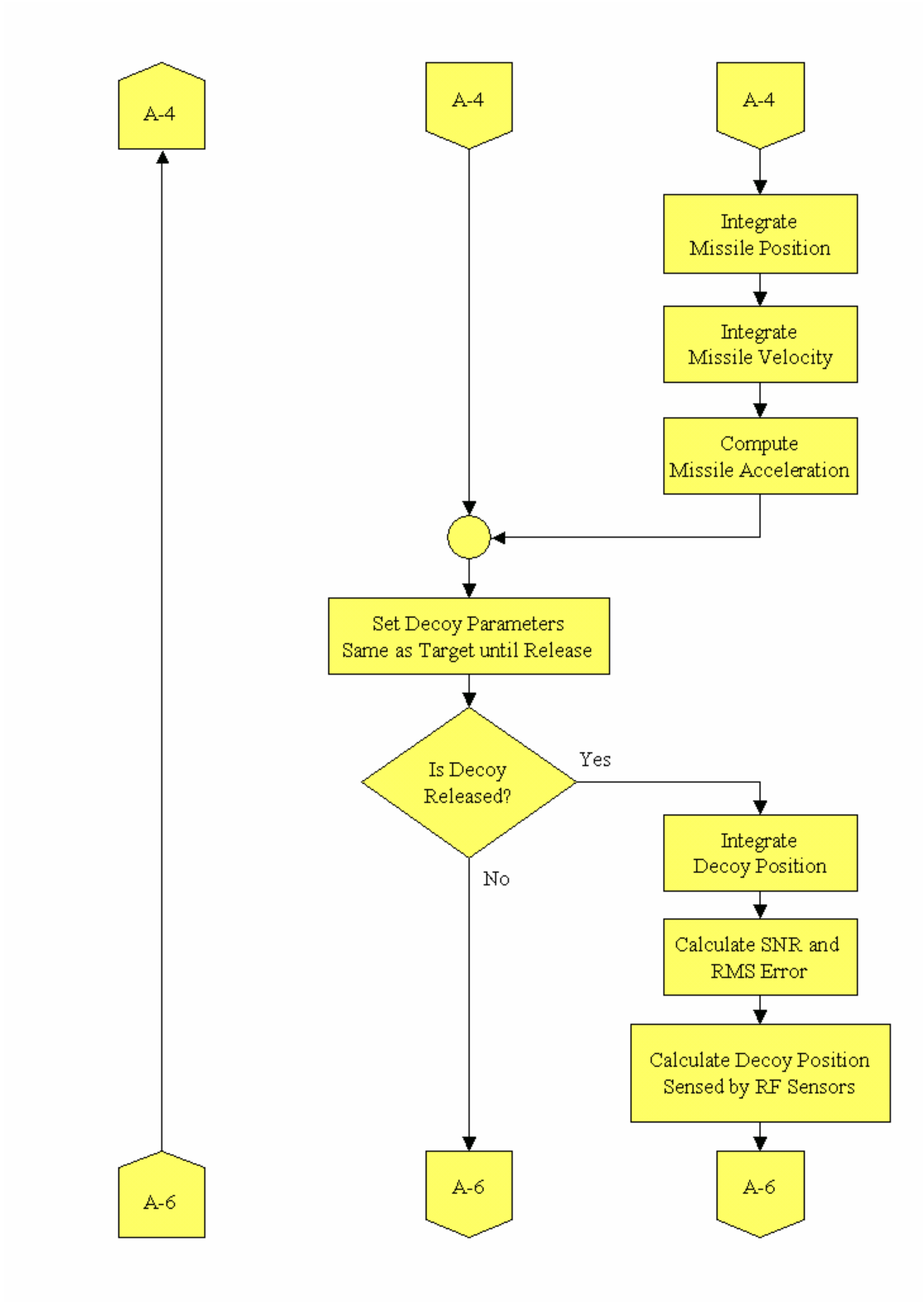

A-4
A-4
A-4
Integrate Missile Position
Integrate Missile Velocity
Compute Missile Acceleration
Set Decoy Parameters Same as Target until Release
Is Decoy Released?
Yes
No
Integrate Decoy Position
Calculate SNR and RMS Error
Calculate Decoy Position Sensed by RF Sensors
A-6
A-6
A-6

Figure A–5. Code Flowchart (5 of 7).

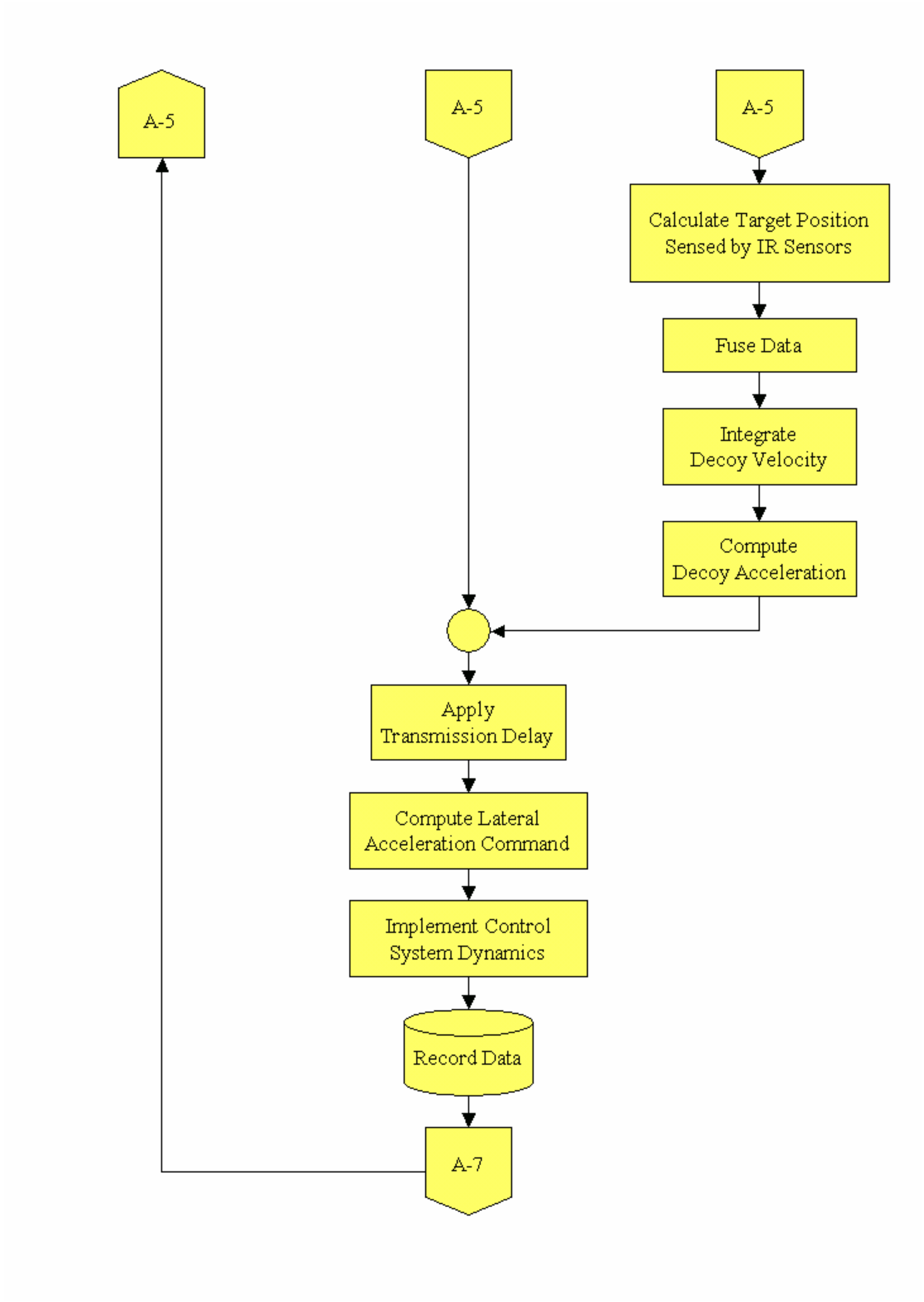

A-5
A-5
A-5
Calculate Target Position Sensed by IR Sensors
Fuse Data
Integrate Decoy Velocity
Compute Decoy Acceleration
Apply Transmission Delay
Compute Lateral Acceleration Command
Implement Control System Dynamics
Record Data
A-7

Figure A–6. Code Flowchart (6 of 7).

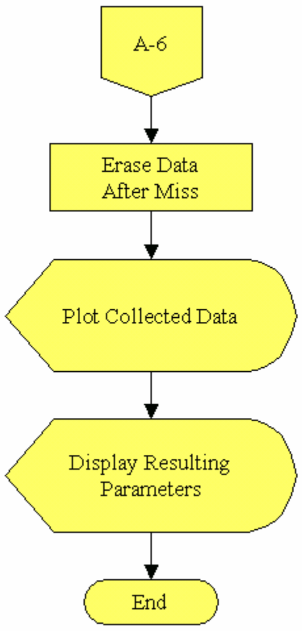


Figure A–7. Code Flowchart (7 of 7).

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B THE MATLAB CODE

This appendix includes a full copy of the MATLAB code, which is developed for this investigation.

```
%3D Simulation for Multi-stage Boosting Target and Missile
%Aug 2 2004, Monterey, CA
%K. Uzun, Prof. P.E. Pace, Prof. M.Tummala

clear;clc;
%-----------------------------------------------------------------------------
%Constants
Re = 6.37e6;        %Radius of the Earth (m)
G = 6.67e-11;       %Gravitational Constant (m^3/s^2.kg)
Me = 5.98e24;       %Earth's Mass (kg)
JE0 = 300e-6;       %Earth's Radiation Intensity (Mixed Terrain) (W/sr)

kT0 = 4e-21;        %Boltzman Constant x 290K
SOL = 3e8;          %Speed of light (m/s)

posEC = [0; 0; 0];  %Position of Earth's Center
mcf = 0.4535924;    %Mass Conversion Factor (lb --> kg)

BIG = 1e6;          %An arbitrary big number for comparisons
SMALL = -1e-6;      %An arbitrary small number for comparisons

ON = 1;             %Boolean switch for ON
OFF = 0;            %Boolean switch for OFF

farTimeStep = 0.05;      %Integration Time Step During Interception Phase (s)
nearTimeStep = 0.0005;   %Integration Time Step at Terminal Phase (s) 0.0005
timeStepSwitch = 5000;   %Distance to Switch Time Step (m)
updateTime = 0.15;       %Sensor Update Interval (s)
txDelay = 10e-3;         %Transmission delay (s)

%-----------------------------------------------------------------------------
%Initialization of Parameters
%Define Missile Data Matrices as Follows
%                           Stage-1     Stage-1     ...     Stage-n
%   Total Mass (lb)       [ X           X           ...     X
%   Propellent Mass (lb)    X           X           ...     X
%   Specific Impulse (s)    X           X           ...     X
%   In-stage Burn Time (s)  X           X           ...     X ]
%Following Rows are Added by the Program (Do not Define!)
%   dM/dt                 [ X           X           ...     X
%   Canister Mass           X           X           ...     X
%   Ignition Time           X           X           ...     X
%   dM/dt Ratio             X           X                   X ]

%Target (Located at Kilju Missile Base, N. Korea)
latHemT = 'N';      %Target Launch Latitude Hemisphere (Either N or S)
latDegT = 41;       %Target Launch Latitude Degree Part (Degrees)
latMinT = 00;       %Target Launch Latitude Minute Part (Minutes)

lonHemT = 'E';      %Target Launch Longitude Hemisphere (Either E or W)
lonDegT = 129;      %Target Launch Longitude Degree Part (Degrees)
lonMinT = 00;       %Target Launch Longitude Minute Part (Minutes)

lAzTd = 50.1;       %Target Launch Angle (Azimuth) (Degrees) (SF: 50.1)
lElTd = 84;         %Target Launch Angle (Elevation) (Degrees) (SF: 84)

lTimeT = 0;         %Target Launch Time (s)

lenT = 21.8;        %Target Total Length (m)

%Load Target RCS Data
load POstage1_V; RCS1VRaw = Sth;    %Stage-1, VHF-Band
load POstage1_L; RCS1LRaw = Sth;    %Stage-1, L-Band
load POstage1_C; RCS1CRaw = Sth;    %Stage-1, C-Band
load POstage1_S; RCS1SRaw = Sth;    %Stage-1, S-Band
load POstage1_X; RCS1XRaw = Sth;    %Stage-1, X-Band
load POstage2_V; RCS2VRaw = Sth;    %Stage-2, VHF-Band
load POstage2_L; RCS2LRaw = Sth;    %Stage-2, L-Band
load POstage2_C; RCS2CRaw = Sth;    %Stage-2, C-Band
load POstage2_S; RCS2SRaw = Sth;    %Stage-2, S-Band
load POstage2_X; RCS2XRaw = Sth;    %Stage-2, X-Band

load POstage3_V; RCS3VRaw = Sth;    %Stage-3, VHF-Band
load POstage3_L; RCS3LRaw = Sth;    %Stage-3, L-Band
load POstage3_C; RCS3CRaw = Sth;    %Stage-3, C-Band
load POstage3_S; RCS3SRaw = Sth;    %Stage-3, S-Band
load POstage3_X; RCS3XRaw = Sth;    %Stage-3, X-Band

load POstage4_V; RCS4VRaw = Sth;    %Glide, VHF-Band
load POstage4_L; RCS4LRaw = Sth;    %Glide, L-Band
load POstage4_C; RCS4CRaw = Sth;    %Glide, C-Band
load POstage4_S; RCS4SRaw = Sth;    %Glide, S-Band
load POstage4_X; RCS4XRaw = Sth;    %Glide, X-Band

%Target Radiation Intensity at Launch
JT0 = 3e6;  %Target Radiation Intensity (W/sr)

%Generic 3-Stage Missile Data Matrix
%               Stage-1     Stage-2     Stage-3     Payload
dataMatrixT = [ 108000      61000       17000       5000
                91800       51850       14450       0
                300         300         300         0
                60          60          60          1   ];

numStageT = size(dataMatrixT, 2);   %Number of Stages (Target)

%Missile (Located at Sea of Japan, 600km East of Target Launch Site)
latHemM = 'N';      %Missile Launch Latitude Hemisphere (Either N or S)
latDegM = 41;       %Missile Launch Latitude Degree Part (Degrees)
latMinM = 00;       %Missile Launch Latitude Minute Part (Minutes)

lonHemM = 'E';      %Missile Launch Longitude Hemisphere (Either E or W)
lonDegM = 136;      %Missile Launch Longitude Degree Part (Degrees)
lonMinM = 09;       %Missile Launch Longitude Minute Part (Minutes)

lAzMd = 298;        %Missile Launch Angle (Azimuth) (Degrees) (271)
lElMd = 79;         %Missile Launch Angle (Elevation) (Degrees) (79)

lTimeM = 0;         %Missile Launch Time (s) 30

lenM = 21.8;        %Missile Total Length (m)

%Generic 3-Stage Interceptor Data Matrix
%               Stage-1     Stage-2     Stage-3     Payload (Kill Vehicle)
dataMatrixM1 = [ 108000      61000       17000       5000
                 91800       51850       14450       0
                 300         300         300         0
                 60          60          60          1   ];

dataMatrixM2 = [ 108000      61000       17000       2500
                 97200       54900       15300       0
                 300         300         300         0
                 60          60          60          1   ];

dataMatrixM3 = [ 108000      61000       17000       1500
                 102600      57950       16150       0
                 300         300         300         0
                 60          60          60          1   ];

dataMatrixM = dataMatrixM3;

navCoefM = 4;       %Missile Navigation Coefficient for Proportional Navigation (Either 3, 4 or 5)

maxG = 10;          %Max Lateral Acceleration Command (g)

%Define Control System Dynamics Transfer Function
%                       1
%           -------------------------
%           as^n+ ... + bs^2 + cs + 1
TMc = 5;                                %System Time Constant
numTFM = [1];                           %Numerator
denTFM = [(TMc^3/27) (TMc^2/3) TMc 1];  %Denominator (3rd Order TF)
sysTFM = tf(numTFM, denTFM);            %Generate Transfer Function From Parameters

sysdTFMFar = c2d(sysTFM,farTimeStep);           %Discretize System (Far)
sysdTFMNear = c2d(sysTFM,nearTimeStep);         %Discretize System (Near)
```

```
[AMFar,BMFar,CMFar,DMFar] = ssdata(sysdTFMFar);      %Convert Transfer Function to State Space (Far)
[AMNear,BMNear,CMNear,DMNear] = ssdata(sysdTFMNear); %Convert Transfer Function to State Space (Near)

numStageM = size(dataMatrixM, 2);   %Number of Stages (Missile)

%Friendly Asset (San Francisco)
latHemFA = 'N';      %Target Launch Latitude Hemisphere (Either N or S)
latDegFA = 37;       %Target Launch Latitude Degree Part (Degrees)
latMinFA = 45;       %Target Launch Latitude Minute Part (Minutes)

lonHemFA = 'W';      %Target Launch Longitude Hemisphere (Either E or W)
lonDegFA = 122;      %Target Launch Longitude Degree Part (Degrees)
lonMinFA = 25;       %Target Launch Longitude Minute Part (Minutes)

%RF-1 (125 degrees 600 km from Target Launch Site, Sea of Japan)
latHemRF1 = 'N';      %RF-1 Latitude Hemisphere (Either N or S)
latDegRF1 = 37;       %RF-1 Latitude Degree Part (Degrees) 35
latMinRF1 = 46;       %RF-1 Latitude Minute Part (Minutes) 36

lonHemRF1 = 'E';      %Target Launch Longitude Hemisphere (Either E or W)
lonDegRF1 = 134;      %Target Launch Longitude Degree Part (Degrees)
lonMinRF1 = 35;       %Target Launch Longitude Minute Part (Minutes)

fRF1 = 10e9;                        %Frequency (Hz)
HPBWRF1 = 0.5;                      %Half Power (3dB) beamwidth (Degrees)
PtRF1 = 1000e3;                     %Peak power (Watts)
GtRF1 = 26000 / (HPBWRF1 * HPBWRF1);%Gain
tauRF1 = 50e-6;                     %Pulsewidth (s)
deltaFRF1 = 1/tauRF1;               %Receiver Bandwidth
PRFRF1 = 150;                       %PRF (Hz)
NiRF1 = 20;                         %# of pulses integrated
FRF1 = 4;                           %Noise factor
lambdaRF1 = SOL / fRF1;             %Wavelength (m)
kmRF1 = 1.7;                        %Constant Given 1.7-1.9 for Monopulse Trackers
HPBWRF1 = HPBWRF1 * pi / 180;       %Half Power (3dB) beamwidth (radians)
FrBsRF1 = NiRF1;                    %Fr/Bs

%RF-2 (135 degrees 600 km from Target Launch Site, Sea of Japan)
latHemRF2 = 'N';      %RF-2 Latitude Hemisphere (Either N or S)
latDegRF2 = 37;       %RF-2 Latitude Degree Part (Degrees)
latMinRF2 = 05;       %RF-2 Latitude Minute Part (Minutes)

lonHemRF2 = 'E';      %RF-2 Longitude Hemisphere (Either E or W)
lonDegRF2 = 133;      %RF-2 Longitude Degree Part (Degrees)
lonMinRF2 = 46;       %RF-2 Longitude Minute Part (Minutes)

fRF2 = 10e9;                        %Frequency (Hz)
HPBWRF2 = 0.5;                      %Half Power (3dB) beamwidth (Degrees)
PtRF2 = 1000e3;                     %Peak power (Watts)
GtRF2 = 26000 / (HPBWRF2 * HPBWRF2);%Gain
tauRF2 = 50e-6;                     %Pulsewidth (s)
deltaFRF2 = 1/tauRF2;               %Receiver Bandwidth
PRFRF2 = 150;                       %PRF (Hz)
NiRF2 = 20;                         %# of pulses integrated
FRF2 = 4;                           %Noise factor
lambdaRF2 = SOL / fRF2;             %Wavelength (m)
kmRF2 = 1.7;                        %Constant Given 1.7-1.9 for Monopulse Trackers
HPBWRF2 = HPBWRF2 * pi / 180;       %Half Power (3dB) beamwidth (radians)
FrBsRF2 = NiRF2;                    %Fr/Bs

%IR-1 (Over Target Launch Location, at 1000km Height)
latHemIR1 = 'N';    %IR-1 Latitude Hemisphere (Either N or S)
latDegIR1 = 41;     %IR-1 Latitude Degree Part (Degrees)
latMinIR1 = 00;     %IR-1 Latitude Minute Part (Minutes)

lonHemIR1 = 'E';    %IR-1 Longitude Hemisphere (Either E or W)
lonDegIR1 = 129;    %IR-1 Longitude Degree Part (Degrees)
lonMinIR1 = 00;     %IR-1 Longitude Minute Part (Minutes)

altIR1 = 1000e3;    %Altitude (m)
IFOVIR1 = 20e-6;    %Instantenous Field of View (IFOV) (rad)
D0IR1 = 50;         %Diameter of Optics (cm)

%IR-2 (Over Missile Launch Location, at 1000km Height)
latHemIR2 = 'N';    %IR-2 Latitude Hemisphere (Either N or S)
latDegIR2 = 41;     %IR-2 Latitude Degree Part (Degrees)
latMinIR2 = 00;     %IR-2 Latitude Minute Part (Minutes)

lonHemIR2 = 'E';    %IR-2 Longitude Hemisphere (Either E or W)
lonDegIR2 = 136;    %IR-2 Longitude Degree Part (Degrees)
lonMinIR2 = 09;     %IR-2 Longitude Minute Part (Minutes)
```

```
altIR2 = 1000e3;    %Altitude (m)
IFOVIR2 = 20e-6;    %Instantenous Field of View (IFOV) (rad)
D0IR2 = 50;         %Diameter of Optics (cm)

%Decoy
%Position = Current Position of target
%Velocity = Current velocity of target
RCSD = 5;           %Decoy Radar Cross Section (dBsm)
JD = 1.5e3;         %Decoy Radiation Intensity (W/sr)
lTimeD = 90;        %Decoy Release Time (s)
MD = 100;           %Decoy Mass (kg)
tReacq = 0;         %Sensor Reacquisition Time (s)

%Jammer
%Position = Current Position of target
PtJ = 1e3  ;        %Jammer Power (W)
deltaFJ = 4e9;      %Jammer Bandwidth (Hz)
jamON = 0;          %Jammer ON/OFF

%-------------------------------------------------------------------------------
%Conversions
%Target
[thetaT phiT] = geoToSph(latHemT, latDegT, latMinT, lonHemT, lonDegT, lonMinT);
%Convert Target Geographic Coordinates to Spherical Coordinates
[xT,yT,zT] = sphToCart(thetaT,phiT,Re);
%Convert Target Spherical Coordinates to Cartesian Coordinates
posT = [xT; yT; zT];
%Target Position Vector
pos0T = posT;
%Target Initial Position

lAzT = lAzTd * pi / 180;    %Target Launch Angle (Azimuth) (Radians)
lElT = lElTd * pi / 180;    %Target Launch Angle (Elevation) (Radians)

%Masses lb --> kg
for r = 1:2
    for c = 1:numStageT
        dataMatrixT(r, c) = dataMatrixT(r, c) * mcf;
    end
end

%Missile
[thetaM phiM] = geoToSph(latHemM, latDegM, latMinM, lonHemM, lonDegM, lonMinM);
%Convert Missile Geographic Coordinates to Spherical Coordinates
[xM,yM,zM] = sphToCart(thetaM,phiM,Re);
%Convert Missile Spherical Coordinates to Cartesian Coordinates
posM = [xM; yM; zM];
%Missile Position Vector
pos0M = posM;
%Missile Initial Position

lAzM = lAzMd * pi / 180;    %Missile Launch Angle (Azimuth) (Radians)
lElM = lElMd * pi / 180;    %Missile Launch Angle (Elevation) (Radians)

%Masses lb --> kg
for r = 1:2
    for c = 1:numStageM
        dataMatrixM(r, c) = dataMatrixM(r, c) * mcf;
    end
end

%Friendly Asset
[thetaFA phiFA] = geoToSph(latHemFA, latDegFA, latMinFA, lonHemFA, lonDegFA, lonMinFA);
%Convert Friendly Asset Geographic Coordinates to Spherical Coordinates
[xFA,yFA,zFA] = sphToCart(thetaFA,phiFA,Re);
%Convert Friendly Asset Spherical Coordinates to Cartesian Coordinates
posFA = [xFA; yFA; zFA];
%Friendly Asset Position Vector

%RF-1
[thetaRF1 phiRF1] = geoToSph(latHemRF1, latDegRF1, latMinRF1, lonHemRF1, lonDegRF1, lonMinRF1);
%Convert RF-1 Geographic Coordinates to Spherical Coordinates
[xRF1,yRF1,zRF1] = sphToCart(thetaRF1,phiRF1,Re);
%Convert RF-1 Spherical Coordinates to Cartesian Coordinates
posRF1 = [xRF1; yRF1; zRF1];
%RF-1 Position Vector

%RF-2
[thetaRF2 phiRF2] = geoToSph(latHemRF2, latDegRF2, latMinRF2, lonHemRF2, lonDegRF2, lonMinRF2);
%Convert RF-2 Geographic Coordinates to Spherical Coordinates
```

```
[xRF2,yRF2,zRF2] = sphToCart(thetaRF2,phiRF2,Re);
%Convert RF-2 Spherical Coordinates to Cartesian Coordinates
posRF2 = [xRF2; yRF2; zRF2];
%RF-2 Position Vector

%IR-1
[thetaIR1 phiIR1] = geoToSph(latHemIR1, latDegIR1, latMinIR1, lonHemIR1, lonDegIR1, lonMinIR1);
%Convert IR-1 Geographic Coordinates to Spherical Coordinates
[xIR1,yIR1,zIR1] = sphToCart(thetaIR1,phiIR1,Re + altIR1);
%Convert IR-1 Spherical Coordinates to Cartesian Coordinates
posIR1 = [xIR1; yIR1; zIR1];
%IR-1 Position Vector

%IR-2
[thetaIR2 phiIR2] = geoToSph(latHemIR2, latDegIR2, latMinIR2, lonHemIR2, lonDegIR2, lonMinIR2);
%Convert IR-2 Geographic Coordinates to Spherical Coordinates
[xIR2,yIR2,zIR2] = sphToCart(thetaIR2,phiIR2,Re + altIR2);
%Convert IR-2 Spherical Coordinates to Cartesian Coordinates
posIR2 = [xIR2; yIR2; zIR2];
%IR-2 Position Vector

%Decoy
posD = posT;    %Decoy Position Vector
pos0D = pos0T;  %Initial Decoy Position Vector
%--------------------------------------------------------------------------------
%Various Computations
g0 = (G * Me) / (Re ^ 2);   %Gravitational Acceleration at Sea Level

%Target
MT = sum(dataMatrixT(1,:)); %Target Initial Total Mass

%Interpolate RCS Data for 0.1 degrees precision
mAngleDegRaw = 0:360;   %Monostatic Angle Theta at Raw Data Precision
mAngleDeg = 0:0.1:360;  %Monostatic Angle Theta after Interpolation

RCS1V = interp1(mAngleDegRaw, RCS1VRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-1, VHF-Band
RCS1L = interp1(mAngleDegRaw, RCS1LRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-1, L-Band
RCS1C = interp1(mAngleDegRaw, RCS1CRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-1, C-Band
RCS1S = interp1(mAngleDegRaw, RCS1SRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-1, S-Band
RCS1X = interp1(mAngleDegRaw, RCS1XRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-1, X-Band

RCS2V = interp1(mAngleDegRaw, RCS2VRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-2, VHF-Band
RCS2L = interp1(mAngleDegRaw, RCS2LRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-2, L-Band
RCS2C = interp1(mAngleDegRaw, RCS2CRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-2, C-Band
RCS2S = interp1(mAngleDegRaw, RCS2SRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-2, S-Band
RCS2X = interp1(mAngleDegRaw, RCS2XRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-2, X-Band

RCS3V = interp1(mAngleDegRaw, RCS3VRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-3, VHF-Band
RCS3L = interp1(mAngleDegRaw, RCS3LRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-3, L-Band
RCS3C = interp1(mAngleDegRaw, RCS3CRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-3, C-Band
RCS3S = interp1(mAngleDegRaw, RCS3SRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-3, S-Band
RCS3X = interp1(mAngleDegRaw, RCS3XRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Stage-3, X-Band

RCS4V = interp1(mAngleDegRaw, RCS4VRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Glide, VHF-Band
RCS4L = interp1(mAngleDegRaw, RCS4LRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Glide, L-Band
RCS4C = interp1(mAngleDegRaw, RCS4CRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Glide, C-Band
RCS4S = interp1(mAngleDegRaw, RCS4SRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Glide, S-Band
RCS4X = interp1(mAngleDegRaw, RCS4XRaw, mAngleDeg);
%Generate 0.1 deg precision RCS Matrix, Glide, X-Band

%Add dM/dt and Canister Weight Rows to Data Matrix
intermVar1 = [];        %Define an intermediate variable
```

```
intermVar2 = [];        %Define an intermediate variable
intermVar3 = [];        %Define an intermediate variable
for c = 1:numStageT     %Loop for the number of stages
    intermVar1 = [intermVar1 (dataMatrixT(2,c) / dataMatrixT(4,c))];    %Generate dM/dt row
    intermVar2 = [intermVar2 (dataMatrixT(1,c) - dataMatrixT(2,c))];    %Generate canister weight row
end
dataMatrixT = [dataMatrixT; intermVar1; intermVar2];    %Reform Matrix
intermVar3 = intermVar1 ./ intermVar1(1);               %dM/dt Ratio

%Add Ignition Time Row to Data Matrix
intermVar1 = cumsum(dataMatrixT(4,:));                  %Sum burn times
intermVar1 = [0 intermVar1(1:(size(intermVar1,2)-1))];  %Generate ignition time row
dataMatrixT = [dataMatrixT; intermVar1;intermVar3];     %Reform Matrix

%Missile
MM = sum(dataMatrixM(1,:)); %Missile Initial Total Mass

%Add dM/dt and Canister Weight Rows to Data Matrix
intermVar1 = [];        %Define an intermediate variable
intermVar2 = [];        %Define an intermediate variable
for c = 1:numStageM     %Loop for the number of stages
    intermVar1 = [intermVar1 (dataMatrixM(2,c) / dataMatrixM(4,c))];    %Generate dM/dt row
    intermVar2 = [intermVar2 (dataMatrixM(1,c) - dataMatrixM(2,c))];    %Generate canister weight row
end
dataMatrixM = [dataMatrixM; intermVar1; intermVar2];    %Reform Matrix

%Add Ignition Time Row to Data Matrix
intermVar1 = cumsum(dataMatrixM(4,:));                  %Sum burn times
intermVar1 = [0 intermVar1(1:(size(intermVar1,2)-1))];  %Generate ignition time row
dataMatrixM = [dataMatrixM; intermVar1];                %Reform Matrix

%-------------------------------------------------------------------------------
%Unit Vector Computations
%Target
unitWT = (posEC - posT) ./ Re;  %Weight Unit Vector

[vxT vyT vzT] = topToCart(lAzT, lElT, latHemT, latDegT, latMinT, lonHemT, lonDegT, lonMinT);
%Velocity Unit Vector
unitvT = [vxT; vyT; vzT];
%Velocity Unit Vector

unitTT = unitvT;                %Thrust Unit Vector

%Missile
unitWM = (posEC - posM) ./ Re;  %Weight Unit Vector

[vxM vyM vzM] = topToCart(lAzM, lElM, latHemM, latDegM, latMinM, lonHemM, lonDegM, lonMinM);
%Velocity Unit Vector
unitvM = [vxM; vyM; vzM];
%Velocity Unit Vector

unitTM = unitvM;                %Thrust Unit Vector

%Decoy
unitWD = (posEC - posD) ./ Re;  %Weight Unit Vector
unitvD = unitvT;                %Velocity Unit Vector
unitTD = unitTT;                %Thrust Unit Vector

%-------------------------------------------------------------------------------
%Simulation
t = 0;              %Simulation Time (s)
dt = farTimeStep;   %Simulation Time Step (s)
AMc = AMFar;        %State Space Matrix-A
BMc = BMFar;        %State Space Matrix-B
CMc = CMFar;        %State Space Matrix-C
DMc = DMFar;        %State Space Matrix-D
flag = 1;           %Flag used to switch between time steps
disp(' ');          %Display a blank line

%Target
stageT = 1;
%Target Stage
stageChangeTimeT = dataMatrixT(7, stageT + 1);
%Target Next Stage Change Time (s)
ISPT = dataMatrixT(3, stageT);
%Target Stage Specific Impulse (s)
dMdtT = dataMatrixT(5, stageT);
%Target Stage dM/dT (kg/s)
hT = 0;
%Target Height (m)
aT = [0; 0; 0];
```

```
%Target Acceleration Vector
magTT = dMdtT * g0 * ISPT;
%Target Stage Sea Level Thrust (N)
magvT = (magTT - MT * g0) / MT * (lenT * MT / (magTT - MT * g0)) ^ (1 / 2);
%Target Silo Exit Velocity (m/s)
vT = magvT .* unitvT;
%Target Velocity Vector
groundTrackT = posT;
%Target Ground Track Vector
distT = 0;                                                  %Target Ground Distance (m)
mnvrT = 0;                                                  %Target Maneuver (g)
sensedPosT = posT;                                          %Target Sensed Position Vector
RCS1 = 1;                                                   %Target RCS seen by RF-1
RCS2 = 1;                                                   %Target RCS seen by RF-2
JT = 0;                                                     %Target Radiation Intensity

%Missile
stageM = 1;                                                 %Missile Stage
stageChangeTimeM = dataMatrixM(7, stageM + 1);              %Missile Next Stage Change Time (s)
ISPM = dataMatrixM(3, stageM);                              %Missile Stage Specific Impulse (s)
dMdtM = dataMatrixM(5, stageM);                             %Missile Stage dM/dT (kg/s)
hM = 0;                                                     %Missile Height (m)
aM = [0; 0; 0];                                             %Missile Acceleration Vector
magTM = dMdtM * g0 * ISPM;                                  %Missile Stage Sea Level Thrust (N)
magvM = (magTM - MM * g0) / MM * (lenM * MM / (magTM - MM * g0)) ^ (1 / 2);
%Missile Silo Exit Velocity (m/s)
vM = magvM .* unitvM;                                   %Missile Velocity Vector
groundTrackM = posM;                                    %Missile Ground Track Vector
distM = 0;                                              %Missile Ground Distance (m)
GFM = [0;0;0];                                          %Missile Guidance Force (N)
comLatAccM = 0;                                         %Commanded Missile Lateral Acceleration (g)
achLatAccM = 0;                                         %Achieved Missile Lateral Acceleration (g)
latDivM = 0;                                            %Missile Lateral Divert (m/s)
nlM = [0; 0; 0];                                        %Achieved Lateral Acceleration Vector
nlPerM = [0;0;0];                  %Achieved Lateral Acceleration Vector Perpendicular Component
stateMX = [0; 0; 0];               %State Matrix Initial Condition, x-axis
stateMY = [0; 0; 0];               %State Matrix Initial Condition, y-axis
stateMZ = [0; 0; 0];               %State Matrix Initial Condition, z-axis
magGFM =0;                         %Magnitude of Guidance Force (N)

%Decoy
hD = 0;                                                %Decoy Height (m)
aD = [0; 0; 0];                                        %Decoy Acceleration Vector
magTD = 0;                                             %Decoy Thrust = 0 (N)
magvD = magvT;                                         %Decoy Initial Velocity (m/s)
vD = magvD .* unitvD;                                  %Decoy Velocity Vector
groundTrackD = posD;                                   %Decoy Ground Track Vector
distD = 0;                                             %Decoy Ground Distance (m)
sensedPosD = posD;                                     %Decoy Sensed Position Vector
oldGroundTrackD = posT;                                %Decoy previous ground track

%Target-Decoy
distTD = 0;                                            %Target-Decoy Distance

%Missile-Target
VcMT = 0;                             %Missile-to-Target Sensed Closing Velocity (m/s)
VcMTTrue = 0;                         %Missile-to-Target True Closing Velocity (m/s)
distMT = 0;                           %Missile-to-Target Distance (m)
LOSRateMT = [0; 0; 0];                %LOS Rate Vector
unitLOSRateMT = [0; 0; 0];            %LOS Rate Unit Vector
oldDistMTTrue = 0;                    %Previous True Missile-to-Target Distance (m)
oldLOSMT = posT - posM;               %Previous Line of Sight (LOS) Between Missile and Target
oldDistMT = sqrt(oldLOSMT(1) ^ 2 + oldLOSMT(2) ^ 2 + oldLOSMT(3) ^ 2);
%Previous Target-Missile Distance


%Sensor
distRF1T = 0;                %RF-1-Target distance
distRF2T = 0;                %RF-2-Target distance
SNR1dB = 0;                  %SNR seen by RF-1 in dB
SNR2dB = 0;                  %SNR seen by RF-2 in dB
SJR1dB = 0;                  %Radiation intensity seen by IR-1
SJR2dB = 0;                  %Radiation intensity seen by IR-2
sigmaAlfaRF1 = 0;            %RMS angular error introduced by RF-1
sigmaAlfaRF2 = 0;            %RMS angular error introduced by RF-2
sigmaAlfaRRF1 = 0;           %RMS angular error introduced by RF-1 at target range
sigmaAlfaRRF2 = 0;           %RMS angular error introduced by RF-2 at target range
sigmaRRF1 = 0;               %RMS range error introduced by RF-1
sigmaRRF2 = 0;               %RMS range error introduced by RF-2
magErrPosTRF1 = 0;           %Magnitude of error introduced by RF-1
magErrPosTRF2 = 0;           %Magnitude of error introduced by RF-2
```

```
magErrPosTIR = 0;            %Magnitude of error introduced by IR sensors
magPosError = 0;             %Magnitude of target position error
txCounter = 0;               %A counter used by the code
txFlag = 1;                  %A flag used by the code
SCRIR1dB = 0;                %Signal-to-clutter ratio IR-1
SCRIR2dB = 0;                %Signal-to-clutter ratio IR-2
lookRF1 = 0;                 %Target aspect angle seen by RF-1
lookRF2 = 0;                 %Target aspect angle seen by RF-2

%Data Recording Arrays
tArray   = [];               %Simulation Time

%Target
posArrayT = [];              %Target Position
sensedPosArrayT = [];        %Sensed Target Position
groundTrackArrayT = [];      %Target Ground Track
distArrayT = [];             %Target Ground Distance Array
hArrayT  = [];               %Target Height (m)
vArrayT  = [];               %Target Velocity (m/s)
stageArrayT = [];            %Target Stage
massArrayT = [];             %Target Total Mass
mnvrArrayT = [];             %Target Maneuver
JTArray = [];                %Target radiation intensity array

%Missile
posArrayM = [];              %Missile Position
groundTrackArrayM = [];      %Missile Ground Track
distArrayM = [];             %Missile Ground Distance Array
hArrayM  = [];               %Missile Height (m)
vArrayM  = [];               %Missile Velocity (m/s)
stageArrayM = [];            %Missile Stage
massArrayM = [];             %Missile Total Mass
comLatAccArrayM = [];        %Commanded Missile Lateral Acceleration (G)
achLatAccArrayM = [];        %Achieved Missile Lateral Acceleration (G)
latDivArrayM = [];           %Missile Lateral Divert (m/s)
VcFlag = 1;

%Decoy
posArrayD = [];              %Decoy Position
sensedPosArrayD = [];        %Sensed Decoy Position
groundTrackArrayD = [];      %Decoy Ground Track
distArrayD = [];             %Decoy Ground Distance Array
hArrayD  = [];               %Decoy Height (m)
vArrayD  = [];               %Decoy Velocity (m/s)

%Target-Decoy
distTDArray = [];

%Missile-Target
distArrayMT = [];            %Missile-Target Distance
VcArrayMT = [];              %Missile-Target Closure Velocity

%Sensor
RCS1Array = [];              %Target RCS Seen by RF-1
RCS2Array = [];              %Target RCS Seen by RF-2
distRF1TArray = [];          %RF-1-Target distance
distRF2TArray = [];          %RF-2-Target distance
SNR1Array = [];              %SNR seen by RF-1
SNR2Array = [];              %SNR seen by RF-2
SJR1Array = [];              %SJR (signal-to-jam ratio) seen by RF-1
SJR2Array = [];              %SJR (signal-to-jam ratio) seen by RF-2
sigmaAlfaRF1Array = [];      %RMS angular error introduced by RF-1
sigmaAlfaRF2Array = [];      %RMS angular error introduced by RF-2
sigmaAlfaRRF1Array = [];     %RMS angular error introduced by RF-1 at target range
sigmaAlfaRRF2Array = [];     %RMS angular error introduced by RF-2 at target range
sigmaRRF1Array = [];         %RMS range error introduced by RF-1
sigmaRRF2Array = [];         %RMS range error introduced by RF-2
magErrPosTRF1Array = [];     %Magnitude of target position error introduced by RF-1
magErrPosTRF2Array = [];     %Magnitude of target position error introduced by RF-2
magErrPosTIRArray = [];      %Magnitude of target position error introduced by IR sensors
magPosErrorArray = [];       %Magnitude of target position error
SCRIR1dBArray = [];          %SCR (Signal-to-clutter) ratio seen by IR-1
SCRIR2dBArray = [];          %SCR (Signal-to-clutter) ratio seen by IR-2
lookRF1Array = [];           %Target aspect angle seen by RF-1
lookRF2Array = [];           %Target aspect angle seen by RF-2

while ((hT >= SMALL) | (hM >= SMALL)) & ((VcMTTrue >= 0) | (t < 90))        %Main loop
    if (distMT <= timeStepSwitch) & flag & (t > dt)
%Check for time step to change state space matrices
        dt = nearTimeStep;  %Change Time steps for precision
        AMc = AMNear;       %State Space Matrix-A
        BMc = BMNear;       %State Space Matrix-B
        CMc = CMNear;       %State Space Matrix-C
        DMc = DMNear;       %State Space Matrix-D
        flag = 0;           %Reset Flag not to Enter Here Again
    end %if (distMT <= timeStepSwitch) & flag & (t > dt)

    if (t >= lTimeT) &  (hT >= SMALL) %When Target Launched (Target Computations)

        %Handle Target Stage Change Computations
        if t >= (stageChangeTimeT + lTimeT)
            disp(['Target Stage-' num2str(stageT) ' Burnout: Speed=' num2str(magvT/1000) ' km/s, Alti-
tude=' num2str(hT / 1000) ' km.']);    %Display Data on stage change
            MT = MT - dataMatrixT(6, stageT);       %Jettison Canister of Previous Stage
            stageT = stageT + 1;                    %Increase Stage Number by One
            ISPT = dataMatrixT(3, stageT);          %Change ISP
            dMdtT = dataMatrixT(5, stageT);         %Change dM/dT
            if stageT >= numStageT                  %If No Next Stage
                stageChangeTimeT = BIG;             %Set Next Stage Change Time to a Huge Number
            else %else stageT >= numStageT
                stageChangeTimeT = dataMatrixT(7, stageT + 1);  %Set Next Stage Change Time
            end %stageT >= numStageT
        end %t >= (stageChangeTimeT + lTimeT)

        magPosT = sqrt(posT(1) ^ 2 + posT(2) ^ 2 + posT(3) ^ 2);    %Magnitude of Position Vector
        unitPosT = posT / magPosT;                                  %Position Unit Vector
        groundTrackT = unitPosT * Re;                               %Groundtrack Vector
        hT =  magPosT - Re;                                         %Height
        gT = (G * Me) / (magPosT ^ 2);                              %Gravitational Acceleration (g)

        if t > (lTimeT + dt)        %Position Integration
            posT = posT + vT * dt;

            %Integrate target downrange
            currDistT = groundTrackT - oldGroundTrackT;
            magDistT = sqrt(currDistT(1) ^ 2 + currDistT(2) ^ 2 + currDistT(3) ^ 2);
            distT = distT + magDistT;

            %RF sensor look angle calculations
            LOSRF1T = posT - posRF1;
%RF-1 --> Target Vector
            magLOSRF1T = sqrt(LOSRF1T(1) ^ 2 + LOSRF1T(2) ^ 2 + LOSRF1T(3) ^ 2);
%Magnitude of RF-1 --> Target Vector
            unitLOSRF1T = LOSRF1T / magLOSRF1T;
%RF-1 --> Target Unit Vector
            lookRF1 = acos(dot(unitLOSRF1T, unitvT));
%RF-1 Look Angle (rad)

            LOSRF2T = posT - posRF2;
%RF-2 --> Target Vector
            magLOSRF2T = sqrt(LOSRF2T(1) ^ 2 + LOSRF2T(2) ^ 2 + LOSRF2T(3) ^ 2);
%Magnitude of RF-2 --> Target Vector
            unitLOSRF2T = LOSRF2T / magLOSRF2T;
%RF-2 --> Target Unit Vector
            lookRF2 = acos(dot(unitLOSRF2T, unitvT));
%RF-2 Look Angle (rad)

            RF1RCSIndex = round((lookRF1*180/pi) * 10) + 1;
%RCS index for RF1
            RF2RCSIndex = round((lookRF2*180/pi) * 10) + 1;
%RCS index for RF2

            %Determine RCS Seen by RF Sensors According to Stage
            if stageT == 1
                RCS1 = RCS1X(RF1RCSIndex);
                RCS2 = RCS1X(RF2RCSIndex);
            elseif stageT == 2
                RCS1 = RCS2X(RF1RCSIndex);
                RCS2 = RCS2X(RF2RCSIndex);
            elseif stageT == 3
                RCS1 = RCS3X(RF1RCSIndex);
                RCS2 = RCS3X(RF2RCSIndex);
            else
                RCS1 = RCS4X(RF1RCSIndex);
                RCS2 = RCS4X(RF2RCSIndex);
            end

            %Convert RCS values : dBsm --> sm
            RCS1 = 10 ^ (RCS1 / 10);
            RCS2 = 10 ^ (RCS2 / 10);
            distRF1T = magLOSRF1T;
            distRF2T = magLOSRF2T;
            %Sensor Computations
            %RF-1
            SNR1 = (PtRF1 .* GtRF1 .^ 2 .* tauRF1 .* 1 .* RCS1 .* lambdaRF1 .^ 2) ./ ((4 .* pi) .^ 3
.* kT0 .* FRF1 .* distRF1T .^ 4);  %Calculate SNR
            SNR1dB = 10 * log10(SNR1);  %Convert to dB

            SJR1 = (PtRF1 .* GtRF1 .* RCS1 .* deltaFJ) ./ ((4 .*pi) .* PtJ .* distRF1T .^ 2 .* del-
taFRF1);  %Calculate SJR
            SJR1dB = 10 .* log10(SJR1); %Convert to dB

            if jamON            %If Jamming exists
                SNR1 = SJR1;    %Replace SNR with SJR (Ignore noise)
            end %if jamON

            sigmaAlfaRF1 = HPBWRF1 ./ (kmRF1 .* sqrt(2 .* SNR1 .* FrBsRF1));
%Angular Error Std. Deviation (rad)
            sigmaRRF1 = (SOL .* tauRF1 ./ 2) ./ (kmRF1 .* sqrt(2 .* SNR1 .* FrBsRF1));
%Range Error Std. Deviation (m)

            sigmaAlfaRRF1 = sigmaAlfaRF1 * distRF1T;    %RMS angular error at target range (m)

            [thetaLOSRF1T,phiLOSRF1T,rLOSRF1T] = CART2SPH(LOSRF1T(1),LOSRF1T(2),LOSRF1T(3));
%Convert Sensor-to-Target Vector to Spherical Coordinates
            deltaRRF1= normrnd(0, sigmaRRF1);           %Generate a Gaussian Range Error
            deltaThetaRF1 = normrnd(0, sigmaAlfaRF1);   %Generate a Gaussian Angle Error for Theta
            deltaPhiRF1 = normrnd(0, sigmaAlfaRF1);     %Generate a Gaussian Angle Error for Phi

            thetaLOSRF1T = thetaLOSRF1T + deltaThetaRF1;%Add Generated Theta Angle Error to Real Theta
            phiLOSRF1T = phiLOSRF1T + deltaPhiRF1;      %Add Generated Phi Angle Error to Real Phi
            rLOSRF1T = rLOSRF1T + deltaRRF1;            %Add Generated Range Error to Real Range

            [xSensedLOSRF1T,ySensedLOSRF1T,zSensedLOSRF1T] =
SPH2CART(thetaLOSRF1T,phiLOSRF1T,rLOSRF1T);    %Convert Sensed LOS Vector to Cartesian Coordinates

            sensedLOSRF1T = [xSensedLOSRF1T;ySensedLOSRF1T;zSensedLOSRF1T];
%Add LOS Vector to RF Sensor Position to Get Sensed Target Position
            sensedPosTRF1 = posRF1 + sensedLOSRF1T;         %Sensed Target Position by RF Sensor

            errPosTRF1 = posT - sensedPosTRF1;
%Error Vector between Sensed and Actual Target Position
            magErrPosTRF1 = sqrt(errPosTRF1(1) ^ 2 + errPosTRF1(2) ^ 2 + errPosTRF1(3) ^ 2);
%Magnitude of Error Vector between Sensed and Actual Target Position

            %RF-2
            SNR2 = (PtRF2 .* GtRF2 .^ 2 .* tauRF2 .* 1 .* RCS2 .* lambdaRF2 .^ 2) ./ ((4 .* pi) .^ 3
.* kT0 .* FRF2 .* distRF2T .^ 4); %Calculate SNR
            SNR2dB = 10 * log10(SNR2);  %Convert to dB

            SJR2 = (PtRF2 .* GtRF2 .* RCS2 .* deltaFJ) ./ ((4 .*pi) .* PtJ .* distRF2T .^ 2 .* del-
taFRF2);  %Calculate SJR
            SJR2dB = 10 .* log10(SJR2); %Convert to dB

            if jamON            %If Jamming exists
                SNR2 = SJR2;    %Replace SNR with SJR (Ignore noise)
            end %if jamON

            sigmaAlfaRF2 = HPBWRF2 ./ (kmRF2 .* sqrt(2 .* SNR2 .* FrBsRF2));
%Angular Error Std. Deviation (rad)
            sigmaRRF2 = (SOL .* tauRF2 ./ 2) ./ (kmRF2 .* sqrt(2 .* SNR2 .* FrBsRF2));
%Range Error Std. Deviation (m)

            sigmaAlfaRRF2 = sigmaAlfaRF2 * distRF2T;    %RMS angular error at target range (m)

            [thetaLOSRF2T,phiLOSRF2T,rLOSRF2T] = CART2SPH(LOSRF2T(1),LOSRF2T(2),LOSRF2T(3));
%Convert Sensor-to-Target Vector to Spherical Coordinates
            deltaRRF2= normrnd(0, sigmaRRF2);        %Generate a Gaussian Range Error
            deltaThetaRF2 = normrnd(0, sigmaAlfaRF2);%Generate a Gaussian Angle Error for Theta
            deltaPhiRF2 = normrnd(0, sigmaAlfaRF2);  %Generate a Gaussian Angle Error for Phi

            thetaLOSRF2T = thetaLOSRF2T + deltaThetaRF2;%Add Generated Theta Angle Error to Real Theta
            phiLOSRF2T = phiLOSRF2T + deltaPhiRF2;      %Add Generated Phi Angle Error to Real Phi
            rLOSRF2T = rLOSRF2T + deltaRRF2;            %Add Generated Range Error to Real Range

            [xSensedLOSRF2T,ySensedLOSRF2T,zSensedLOSRF2T] =
SPH2CART(thetaLOSRF2T,phiLOSRF2T,rLOSRF2T);    %Convert Sensed LOS Vector to Cartesian Coordinates

            sensedLOSRF2T = [xSensedLOSRF2T;ySensedLOSRF2T;zSensedLOSRF2T];
%Add LOS Vector to RF Sensor Position to Get Sensed Target Position
            sensedPosTRF2 = posRF2 + sensedLOSRF2T;         %Sensed Target Position by RF Sensor
            errPosTRF2 = posT - sensedPosTRF2;
%Error Vector between Sensed and Actual Target Position
            magErrPosTRF2 = sqrt(errPosTRF2(1) ^ 2 + errPosTRF2(2) ^ 2 + errPosTRF2(3) ^ 2);
%Magnitude of Error Vector between Sensed and Actual Target Position

            %IR sensors
            JT = JT0 * dataMatrixT(8, stageT);  %Downscale radiation intensity proportional to dM/dt

            %IR-1
            LOSIR1T = posT - posIR1;
%IR-1 --> Target Vector
            magLOSIR1T = sqrt(LOSIR1T(1) ^ 2 + LOSIR1T(2) ^ 2 + LOSIR1T(3) ^ 2);
%Magnitude of IR-1 --> Target Vector
            unitLOSIR1T = LOSIR1T / magLOSIR1T;
%IR-1 --> Target Unit Vector
            [thetaLOSIR1T,phiLOSIR1T,rLOSIR1T] =
CART2SPH(unitLOSIR1T(1),unitLOSIR1T(2),unitLOSIR1T(3));
%Convert Sensor-to-Target Vector to Spherical Coordinates
            errThetaIR1 = IFOVIR1 * (rand - 0.5);      %Generate random position error in theta
            thetaLOSIR1T = thetaLOSIR1T + errThetaIR1; %Add random error to theta
            errPhiIR1 = IFOVIR1 * (rand - 0.5);        %Generate random position error in phi
            phiLOSIR1T = phiLOSIR1T + errPhiIR1;       %Add random error to phi
            [xSensedLOSIR1T,ySensedLOSIR1T,zSensedLOSIR1T] =
SPH2CART(thetaLOSIR1T,phiLOSIR1T,rLOSIR1T);    %Convert Sensed LOS Vector to Cartesian Coordinates
            sensedLOSIR1T = [xSensedLOSIR1T;ySensedLOSIR1T;zSensedLOSIR1T];
%Generate sensed LOS by IR-1

            distIR1T = magLOSIR1T * 100;                %IR-1 Target distance (cm)
            distIR1E = altIR1 * 100;                    %IR-1 Earth Distance (cm)
            JE = JE0 * (distIR1E * IFOVIR1)^2;          %Clutter Radiated from Footprint (W/sr)
            SIR1 = ((pi*D0IR1^2)/4)*(JT/distIR1T^2) ;   %Signal Power at sensor(W)
            CIR1 = ((pi*D0IR1^2)/4)*(JE/distIR1E^2);    %Clutter Power at sensor(W)
            SCRIR1dB = 10*log10(SIR1/CIR1);             %Signal/Clutter Ratio (dB)


            %IR-2
            LOSIR2T = posT - posIR2;
%IR-2 --> Target Vector
            magLOSIR2T = sqrt(LOSIR2T(1) ^ 2 + LOSIR2T(2) ^ 2 + LOSIR2T(3) ^ 2);
%Magnitude of IR-2 --> Target Vector
            unitLOSIR2T = LOSIR2T / magLOSIR2T;            %IR-2 --> Target Unit Vector
            [thetaLOSIR2T,phiLOSIR2T,rLOSIR2T] =
CART2SPH(unitLOSIR2T(1),unitLOSIR2T(2),unitLOSIR2T(3));
%Convert Sensor-to-Target Vector to Spherical Coordinates
            errThetaIR2 = IFOVIR2 * (rand - 0.5);      %Generate random position error in theta
            thetaLOSIR2T = thetaLOSIR2T + errThetaIR2; %Add random error to theta
            errPhiIR2 = IFOVIR2 * (rand - 0.5);        %Generate random position error in phi
            phiLOSIR2T = phiLOSIR2T + errPhiIR2;       %Add random error to phi
            [xSensedLOSIR2T,ySensedLOSIR2T,zSensedLOSIR2T] =
SPH2CART(thetaLOSIR2T,phiLOSIR2T,rLOSIR2T);    %Convert Sensed LOS Vector to Cartesian Coordinates
            sensedLOSIR2T = [xSensedLOSIR2T;ySensedLOSIR2T;zSensedLOSIR2T];
%Generate sensed LOS by IR-1

            distIR2T = magLOSIR2T * 100;                %IR-2 Target distance (cm)
            distIR2E = altIR2 * 100;                    %IR-2 Earth Distance (cm)
            JE = JE0 * (distIR2E * IFOVIR2)^2;          %Clutter Radiated from Footprint (W/sr)
            SIR2 = ((pi*D0IR2^2)/4)*(JT/distIR2T^2);    %Signal Power at sensor(W)
            CIR2 = ((pi*D0IR2^2)/4)*(JE/distIR2E^2);    %Clutter Power at sensor(W)
            SCRIR2dB = 10*log10(SIR2/CIR2);             %Signal/Clutter Ratio (dB)

            %IR-1/2
            LOSIR1IR2 = posIR1 - posIR2;
%IR-1 --> IR-2 Vector
            magLOSIR1IR2 = sqrt(LOSIR1IR2(1) ^ 2 + LOSIR1IR2(2) ^ 2 + LOSIR1IR2(3) ^ 2);
%Magnitude of IR-1 --> IR-2 Vector
            unitLOSIR1IR2 = LOSIR1IR2 / magLOSIR1IR2;
%IR-1 --> IR-2 Unit Vector
            distIR1IR2 = magLOSIR1IR2;
%IR-1 --> IR-2 Distance
            alfaIR1 = acos(dot(unitLOSIR1IR2,sensedLOSIR1T));
%Angle between IR-1/IR-2 line and IR-1/target line
            alfaIR2 = acos(dot(unitLOSIR1IR2,sensedLOSIR2T));
%Angle between IR-1/IR-2 line and IR-2/target line
            %Maintenance Conversions
            if alfaIR1 > (pi/2)
                alfaIR1 = pi - alfaIR1;
            end
            if alfaIR2 > (pi/2)
                alfaIR2 = pi - alfaIR2;
            end
            alfaT = pi - (alfaIR1 + alfaIR2);
            %Apply law-of-sines to generate sensed target position
            distIR1T = distIR1IR2 * sin(alfaIR2) / sin(alfaT);
            distIR2T = distIR1IR2 * sin(alfaIR1) / sin(alfaT);
            sensedPosTIR = posIR1 + unitLOSIR1T * distIR1T;

            posErrorIR = posT - sensedPosTIR;
%Calculate diff. vector between real and sensed target positions by IR sensors
            magErrPosTIR = sqrt(posErrorIR(1) ^ 2 + posErrorIR(2) ^ 2 + posErrorIR(3) ^ 2);
%Calculate magnitude of target position erro introduced by IR sensors

            %Fusion Box (Average of Inputs)
            sensedPosT = (sensedPosTRF1 + sensedPosTRF2 + sensedPosTIR) ./ 3;
%Fuse (Average) target position data
            posError = posT - sensedPosT;
%Calculate target position error vector
            magPosError = sqrt(posError(1) ^ 2 + posError(2) ^ 2 + posError(3) ^ 2);
%Calculate magnitude of target position error (fused)

        end %t > (lTimeT + dt) (Position Integration)

        if t > lTimeT    %Velocity Integration
            vT = vT + aT * dt;
            magvT = sqrt(vT(1) ^ 2 + vT(2) ^ 2 + vT(3) ^ 2);
            unitvT = vT / magvT;
        end %t > lTimeT (Velocity Integration)

        %Force & Acceleration Computations
        magWT = MT * gT;                %Magnitude of Weight Vector
        unitWT = -unitPosT;             %Weight Unit Vector
        WT = magWT * unitWT;            %Weight Vector

        magTT = dMdtT * gT * ISPT;      %Magnitude Thrust Vector
        unitTT = unitvT;                %Thrust Unit Vector
        TT = magTT * unitTT;            %Thrust Vector

        netForceT = TT + WT;            %Net Force
        aT = netForceT / MT;            %Net Acceleration

        oldGroundTrackT = groundTrackT; %Record ground track
        MT = MT - dMdtT * dt;           %Decrease Current Mass

    end %(t >= lTimeT) &  (hT >= SMALL) (When Target Launched)

    if (t >= lTimeM) & (hM >= SMALL)  %When Missile Launched

        %Handle Missile Stage Change Computations
        if t >= (stageChangeTimeM + lTimeM)
            disp(['Missile Stage-' num2str(stageM) ' Burnout: Speed=' num2str(magvM/1000) ' km/s, Al-
titude=' num2str(hM / 1000) ' km.']);   %Display Data on stage change
            MM = MM - dataMatrixM(6, stageM);       %Jettison Canister of Previous Stage
            stageM = stageM + 1;                    %Increase Stage Number by One
            ISPM = dataMatrixM(3, stageM);          %Change ISP
            dMdtM = dataMatrixM(5, stageM);         %Change dM/dt
            if stageM >= numStageM                  %If No Next Stage
                stageChangeTimeM = BIG;             %Set Next Stage Change Time to a Huge Number
            else                                    %Else
                stageChangeTimeM = dataMatrixM(7, stageM + 1);  %Set Next Stage Change Time
            end
        end

        magPosM = sqrt(posM(1) ^ 2 + posM(2) ^ 2 + posM(3) ^ 2);    %Magnitude of Position Vector
        unitPosM = posM / magPosM;                                  %Position Unit Vector
        groundTrackM = unitPosM * Re;                               %Groundtrack Vector
        hM =  magPosM - Re;                                         %Height
        gM = (G * Me) / (magPosM ^ 2);                              %Gravitational Acceleration (g)

        if t > (lTimeM + dt)   %Position Integration
            posM = posM + vM * dt;

            %Integrate missile downrange
            currDistM = groundTrackM - oldGroundTrackM;
            magDistM = sqrt(currDistM(1) ^ 2 + currDistM(2) ^ 2 + currDistM(3) ^ 2);
            distM = distM + magDistM;
        end

        if t > lTimeM    %Velocity Integration
            vM = vM + aM * dt;
            magvM = sqrt(vM(1) ^ 2 + vM(2) ^ 2 + vM(3) ^ 2);
            unitvM = vM / magvM;
        end
        %Force & Acceleration Computations
        magWM = MM * gM;                %Magnitude of Weight Vector
        unitWM = -unitPosM;             %Weight Unit Vector
        WM = magWM * unitWM;            %Weight Vector

        magTM = dMdtM * gM * ISPM;      %Magnitude Thrust Vector
        %Compansate for Guidance Command
        if (magGFM >= magTM)                        %If Lateral Acceleration Requirements Exceeds
Available Thrust
            magGFM = magTM;                         %Apply as Much as Available
            magTM = 0;                              %Set Thrust to zero
        else
            magTM = sqrt(magTM^2 - magGFM^2);       %If not, compansate Thrust for the Guidance Force
        end
        unitTM = unitvM;                            %Thrust Unit Vector
        TM = magTM * unitTM;                        %Thrust Vector

        netForceM = TM + WM + GFM;                  %Net Force

        aM = netForceM / MM;                        %Net Acceleration

        oldGroundTrackM = groundTrackM;             %Record previous groundtrack
        MM = MM - dMdtM * dt;                       %Decrease Current Mass

    end %(t >= lTimeM) & (hM >= SMALL) (When Missile Launched)

    if (t < lTimeD) %Until Decoy is Released it moves with the target
        posD = posT;
        magPosD = sqrt(posD(1) ^ 2 + posD(2) ^ 2 + posD(3) ^ 2);    %Magnitude of Position Vector
        unitPosD = posD / magPosD;                                  %Position Unit Vector
        groundTrackD = unitPosD * Re;                               %Groundtrack Vector
        hD =  magPosD - Re;                                         %Height
        %Integrate decoy downrange
        currDistD = groundTrackD - oldGroundTrackD;
        magDistD = sqrt(currDistD(1) ^ 2 + currDistD(2) ^ 2 + currDistD(3) ^ 2);
        distD = distD + magDistD;
        %Set decoy speed to target speed until it's released
        vD = vT;
        magvD = sqrt(vD(1) ^ 2 + vD(2) ^ 2 + vD(3) ^ 2);
        oldGroundTrackD = groundTrackD;
    end %(t < lTimeD) (Until decoy is released)

    if (t >= lTimeD) &  (hD >= SMALL) %When Decoy Released

        magPosD = sqrt(posD(1) ^ 2 + posD(2) ^ 2 + posD(3) ^ 2);    %Magnitude of Position Vector
        unitPosD = posD / magPosD;                                  %Position Unit Vector
        groundTrackD = unitPosD * Re;                               %Groundtrack Vector
        hD =  magPosD - Re;                                         %Height
        gD = (G * Me) / (magPosD ^ 2);                              %Gravitational Acceleration (g)

        if t > (lTimeD + dt)   %Position Integration
            posD = posD + vD * dt;

            %Integrate decoy downrange
            currDistD = groundTrackD - oldGroundTrackD;
            magDistD = sqrt(currDistD(1) ^ 2 + currDistD(2) ^ 2 + currDistD(3) ^ 2);
            distD = distD + magDistD;

            %RF sensor look angle calculations
            LOSRF1D = posD - posRF1;
%RF-1 --> Decoy Vector
            magLOSRF1D = sqrt(LOSRF1D(1) ^ 2 + LOSRF1D(2) ^ 2 + LOSRF1D(3) ^ 2);
%Magnitude of RF-1 --> Decoy Vector
            unitLOSRF1D = LOSRF1D / magLOSRF1D;
%RF-1 --> Decoy Unit Vector

            LOSRF2D = posD - posRF2;
%RF-2 --> Decoy Vector
            magLOSRF2D = sqrt(LOSRF2D(1) ^ 2 + LOSRF2D(2) ^ 2 + LOSRF2D(3) ^ 2);
%Magnitude of RF-2 --> Decoy Vector
            unitLOSRF2D = LOSRF2D / magLOSRF2D;
%RF-2 --> Decoy Unit Vector

            %Convert RCS values : dBsm --> sm
            RCSD = 10 ^ (RCSD / 10);
            distRF1D = magLOSRF1D;
            distRF2D = magLOSRF2D;

            %Sensor Computations
            %RF-1
            SNR1D = (PtRF1 .* GtRF1 .^ 2 .* tauRF1 .* 1 .* RCSD .* lambdaRF1 .^ 2) ./ ((4 .* pi) .^ 3
.* kT0 .* FRF1 .* distRF1D .^ 4); %Calculate SNR
            SNR1dBD = 10 * log10(SNR1D);    %Convert to dB
            sigmaAlfaRF1D = HPBWRF1 ./ (kmRF1 .* sqrt(2 .* SNR1D .* FrBsRF1));
%Angular Error Std. Deviation (rad)
            sigmaRRF1D = (SOL .* tauRF1 ./ 2) ./ (kmRF1 .* sqrt(2 .* SNR1D .* FrBsRF1));
%Range Error Std. Deviation (m)

            sigmaAlfaRRF1D = sigmaAlfaRF1D * distRF1D;  %RF-1/Decoy RMS error at target range (m)

            [thetaLOSRF1D,phiLOSRF1D,rLOSRF1D] = CART2SPH(LOSRF1D(1),LOSRF1D(2),LOSRF1D(3));
%Convert Sensor-to-Decoy Vector to Spherical Coordinates
            deltaRRF1D= normrnd(0, sigmaRRF1D);         %Generate a Gaussian Range Error
            deltaThetaRF1D = normrnd(0, sigmaAlfaRF1D); %Generate a Gaussian Angle Error for Theta
            deltaPhiRF1D = normrnd(0, sigmaAlfaRF1D);   %Generate a Gaussian Angle Error for Phi

            thetaLOSRF1D = thetaLOSRF1D + deltaThetaRF1D;
%Add Generated Theta Angle Error to Real Theta
            phiLOSRF1D = phiLOSRF1D + deltaPhiRF1D;
%Add Generated Phi Angle Error to Real Phi
            rLOSRF1D = rLOSRF1D + deltaRRF1D;
%Add Generated Range Error to Real Range

            [xSensedLOSRF1D,ySensedLOSRF1D,zSensedLOSRF1D] =
SPH2CART(thetaLOSRF1D,phiLOSRF1D,rLOSRF1D);    %Convert Sensed LOS Vector to Cartesian Coordinates

            sensedLOSRF1D = [xSensedLOSRF1D;ySensedLOSRF1D;zSensedLOSRF1D];
%Add LOS Vector to RF Sensor Position to Get Sensed Decoy Position
            sensedPosDRF1 = posRF1 + sensedLOSRF1D; %Sensed Decoy Position by RF Sensor

            errPosDRF1 = posD - sensedPosDRF1;  %Error Vector between Sensed and Actual Decoy Position
            magErrPosDRF1 = sqrt(errPosDRF1(1) ^ 2 + errPosDRF1(2) ^ 2 + errPosDRF1(3) ^ 2);
%Magnitude of Error Vector between Sensed and Actual Decoy Position

            %RF-2
            SNR2D = (PtRF2 .* GtRF2 .^ 2 .* tauRF2 .* 1 .* RCSD .* lambdaRF2 .^ 2) ./ ((4 .* pi) .^ 3
.* kT0 .* FRF2 .* distRF2D .^ 4); %Calculate SNR
            SNR2dBD = 10 * log10(SNR2D);    %Convert to dB
            sigmaAlfaRF2D = HPBWRF2 ./ (kmRF2 .* sqrt(2 .* SNR2D .* FrBsRF1));
%Angular Error Std. Deviation
            sigmaRRF2D = (SOL .* tauRF2 ./ 2) ./ (kmRF2 .* sqrt(2 .* SNR2D .* FrBsRF1));
%Range Error Std. Deviation

            sigmaAlfaRRF2D = sigmaAlfaRF2D * distRF2D;  %RF-2/Decoy RMS error at target range (m)

            [thetaLOSRF2D,phiLOSRF2D,rLOSRF2D] = CART2SPH(LOSRF2D(1),LOSRF2D(2),LOSRF2D(3));
%Convert Sensor-to-Decoy Vector to Spherical Coordinates
            deltaRRF2D= normrnd(0, sigmaRRF2D);        %Generate a Gaussian Range Error
            deltaThetaRF2D = normrnd(0, sigmaAlfaRF2D);%Generate a Gaussian Angle Error for Theta
            deltaPhiRF2D = normrnd(0, sigmaAlfaRF2D);  %Generate a Gaussian Angle Error for Phi

            thetaLOSRF2D = thetaLOSRF2D + deltaThetaRF2D;
%Add Generated Theta Angle Error to Real Theta
            phiLOSRF2D = phiLOSRF2D + deltaPhiRF2D;    %Add Generated Phi Angle Error to Real Phi
            rLOSRF2D = rLOSRF2D + deltaRRF2D;          %Add Generated Range Error to Real Range

            [xSensedLOSRF2D,ySensedLOSRF2D,zSensedLOSRF2D] =
SPH2CART(thetaLOSRF2D,phiLOSRF2D,rLOSRF2D);    %Convert Sensed LOS Vector to Cartesian Coordinates

            sensedLOSRF2D = [xSensedLOSRF2D;ySensedLOSRF2D;zSensedLOSRF2D];
%Add LOS Vector to RF Sensor Position to Get Sensed Decoy Position
            sensedPosDRF2 = posRF2 + sensedLOSRF2D;         %Sensed Decoy Position by RF Sensor

            errPosDRF2 = posD - sensedPosDRF2;%Error Vector between Sensed and Actual Decoy Position
            magErrPosDRF2 = sqrt(errPosDRF2(1) ^ 2 + errPosDRF2(2) ^ 2 + errPosDRF2(3) ^ 2);    %Mag-
nitude of Error Vector between Sensed and Actual Decoy Position

            %IR sensors

            %IR-1
            LOSIR1D = posD - posIR1;
%IR-1 --> Decoy Vector
            magLOSIR1D = sqrt(LOSIR1D(1) ^ 2 + LOSIR1D(2) ^ 2 + LOSIR1D(3) ^ 2);
%Magnitude of IR-1 --> Decoy Vector
            unitLOSIR1D = LOSIR1D / magLOSIR1D;
%IR-1 --> Decoy Unit Vector
            [thetaLOSIR1D,phiLOSIR1D,rLOSIR1D] =
CART2SPH(unitLOSIR1D(1),unitLOSIR1D(2),unitLOSIR1D(3));
%Convert Sensor-to-Decoy Vector to Spherical Coordinates
            errThetaIR1D = IFOVIR1 * (rand - 0.5);
%Generate random error in theta
```

```
            thetaLOSIR1D = thetaLOSIR1D + errThetaIR1D;
%Add random error to theta
            errPhiIR1D = IFOVIR1 * (rand - 0.5);
%Generate random error in phi
            phiLOSIR1D = phiLOSIR1D + errPhiIR1D;
%Add random erro to phi
            [xSensedLOSIR1D,ySensedLOSIR1D,zSensedLOSIR1D] =
SPH2CART(thetaLOSIR1D,phiLOSIR1D,rLOSIR1D);    %Convert Sensed LOS Vector to Cartesian Coordinates
            sensedLOSIR1D = [xSensedLOSIR1D;ySensedLOSIR1D;zSensedLOSIR1D];
%Generate sensed decoy position matrix

            distIR1D = magLOSIR1D * 100;                   %IR-1 Decoy distance (cm)
            distIR1E = altIR1 * 100;                       %IR-1 Earth Distance (cm)
            JE = JE0 * (distIR1E * IFOVIR1)^2;             %Clutter Radiated from Footprint (W/sr)
            SIR1D = ((pi*D0IR1^2)/4)*(JD/distIR1D^2) ;     %Signal Power at sensor(W)
            CIR1D = ((pi*D0IR1^2)/4)*(JE/distIR1E^2);      %Clutter Power at sensor(W)
            SCRIR1dBD = 10*log10(SIR1D/CIR1D);             %Signal/Clutter Ratio (dB)


            %IR-2
            LOSIR2D = posD - posIR2;
%IR-2 --> Decoy Vector
            magLOSIR2D = sqrt(LOSIR2D(1) ^ 2 + LOSIR2D(2) ^ 2 + LOSIR2D(3) ^ 2);
%Magnitude of IR-2 --> Decoy Vector
            unitLOSIR2D = LOSIR2D / magLOSIR2D;
%IR-2 --> Decoy Unit Vector
            [thetaLOSIR2D,phiLOSIR2D,rLOSIR2D] =
CART2SPH(unitLOSIR2D(1),unitLOSIR2D(2),unitLOSIR2D(3));
%Convert Sensor-to-Decoy Vector to Spherical Coordinates
            errThetaIR2D = IFOVIR2 * (rand - 0.5);
%Generate random error in theta
            thetaLOSIR2D = thetaLOSIR2D + errThetaIR2D;
%Add random error to theta
            errPhiIR2D = IFOVIR2 * (rand - 0.5);
%Generate random error in phi
            phiLOSIR2D = phiLOSIR2D + errPhiIR2D;
%Add random error to phi
            [xSensedLOSIR2D,ySensedLOSIR2D,zSensedLOSIR2D] =
SPH2CART(thetaLOSIR2D,phiLOSIR2D,rLOSIR2D);    %Convert Sensed LOS Vector to Cartesian Coordinates
            sensedLOSIR2D = [xSensedLOSIR2D;ySensedLOSIR2D;zSensedLOSIR2D];
%Generate sensed decoy position

            distIR2D = magLOSIR2D * 100;                   %IR-2 Decoy distance (cm)
            distIR2E = altIR2 * 100;                       %IR-2 Earth Distance (cm)
            JE = JE0 * (distIR2E * IFOVIR2)^2;             %Clutter Radiated from Footprint (W/sr)
            SIR2D = ((pi*D0IR2^2)/4)*(JD/distIR2D^2);      %Signal Power at sensor(W)
            CIR2D = ((pi*D0IR2^2)/4)*(JE/distIR2E^2);      %Clutter Power at sensor(W)
            SCRIR2dBD = 10*log10(SIR2D/CIR2D);             %Signal/Clutter Ratio (dB)

            %IR-1/2
            LOSIR1IR2 = posIR1 - posIR2;
%IR-1 --> IR-2 Vector
            magLOSIR1IR2 = sqrt(LOSIR1IR2(1) ^ 2 + LOSIR1IR2(2) ^ 2 + LOSIR1IR2(3) ^ 2);
%Magnitude of IR-1 --> IR-2 Vector
            unitLOSIR1IR2 = LOSIR1IR2 / magLOSIR1IR2;
%IR-1 --> IR-2 Unit Vector
            distIR1IR2 = magLOSIR1IR2;
%IR-1 --> IR-2 distance
            alfaIR1D = acos(dot(unitLOSIR1IR2,sensedLOSIR1D));
%Angle between IR-1/IR-2 line and IR-1/Decoy Line
            alfaIR2D = acos(dot(unitLOSIR1IR2,sensedLOSIR2D));
%Angle between IR-1/IR-2 line and IR-2/Decoy Line
            %Maintenance conversions
            if alfaIR1D > (pi/2)
                alfaIR1D = pi - alfaIR1D;
            end
            if alfaIR2D > (pi/2)
                alfaIR2D = pi - alfaIR2D;
            end
            alfaD = pi - (alfaIR1D + alfaIR2D);

            %Use law-of-sines to generate decoy position
            distIR1D = distIR1IR2 * sin(alfaIR2D) / sin(alfaD);
            distIR2D = distIR1IR2 * sin(alfaIR1D) / sin(alfaD);
            sensedPosDIR = posIR1 + unitLOSIR1D * distIR1D;

            posErrorIRD = posD - sensedPosDIR;  %DEcoy position error matrix
            magErrPosDIR = sqrt(posErrorIRD(1) ^ 2 + posErrorIRD(2) ^ 2 + posErrorIRD(3) ^ 2);
%Magnitude of decoy position error

            %Fusion Box (Average of Inputs)
            sensedPosD = (sensedPosDRF1 + sensedPosDRF2 + sensedPosDIR) ./ 3;   %Fuse data
            if t < (lTimeD + tReacq)        %If before reacquisition
                sensedPosT = sensedPosD;    %Trasfer track to decoy
            end
            posErrorD = posD - sensedPosD;  %Decoy position error vector
            magPosErrorD = sqrt(posErrorD(1) ^ 2 + posErrorD(2) ^ 2 + posErrorD(3) ^ 2);
%Magnitude of decoy position error vector
        end %t > (lTimeD + dt) (Position Integration)

        if t > lTimeD    %Velocity Integration
            vD = vD + aD * dt;
            magvD = sqrt(vD(1) ^ 2 + vD(2) ^ 2 + vD(3) ^ 2);
            unitvD = vD / magvD;
        end

        %Force & Acceleration Computations
        magWD = MD * gD;                %Magnitude of Weight Vector
        unitWD = -unitPosD;             %Weight Unit Vector
        WD = magWD * unitWD;            %Weight Vector

        magTD = 0;                      %Magnitude Thrust Vector
        unitTD = unitvD;                %Thrust Unit Vector
        TD = magTD * unitTD;            %Thrust Vector

        netForceD = TD + WD;            %Net Force
        aD = netForceD / MD;            %Net Acceleration

        oldGroundTrackD = groundTrackD; %Record previous ground track

        LOSTD = posT - posD;
        distTD = sqrt(LOSTD(1) ^ 2 + LOSTD(2) ^ 2 + LOSTD(3) ^ 2);

    end %(t >= lTimeD) &  (hD >= SMALL) (When Decoy Released)

    LOSMTTrue = posT - posM;    %True Target-Missile Vector
    distMTTrue = sqrt(LOSMTTrue(1) ^ 2 + LOSMTTrue(2) ^ 2 + LOSMTTrue(3) ^ 2);
%True Target-Missile Distance

    %Apply transmission delay
    receivedPosT = sensedPosT;
    receivedPosT = receivedPosT + (-vT .* txDelay);

    LOSMT = receivedPosT - posM;                %Line of Sight (LOS) Between Missile and Target
    distMT = sqrt(LOSMT(1) ^ 2 + LOSMT(2) ^ 2 + LOSMT(3) ^ 2);          %Target-Missile Distance

     if t > (dt + lTimeM)
        VcMTTrue = (oldDistMTTrue - distMTTrue) / dt;                   %True Closure Velocity (Vc)

        %Compute Control Acceleration at Only Data Update Intervals
        if txFlag | (txCounter >= updateTime)

            VcMT = (oldDistMT - distMT) / updateTime;
%Closure Velocity (Vc)
            %Compute Magnitude and Direction of Lateral Acceleration
            magOldLOSMT = sqrt(oldLOSMT(1) ^ 2 + oldLOSMT(2) ^ 2 + oldLOSMT(3) ^ 2);
%Magnitude of previous LOS
            unitOldLOSMT = oldLOSMT / magOldLOSMT;
%Normalize Previous LOS
            magLOSMT = sqrt(LOSMT(1) ^ 2 + LOSMT(2) ^ 2 + LOSMT(3) ^ 2);
%Magnitude of this LOS
            unitLOSMT = LOSMT / magLOSMT;
%Normalize This LOS
            deltaLOSMT = unitLOSMT - unitOldLOSMT;
%Find LOS Change Direction (=Direction of Lateral Acceleration)
            magDeltaLOSMT = sqrt(deltaLOSMT(1) ^ 2 + deltaLOSMT(2) ^ 2 + deltaLOSMT(3) ^ 2);
%Magnitude of LOS Change
            magLOSRateMT = magDeltaLOSMT / updateTime;
%Magnitude of LOS Rate (rad/s)

            unitncM = deltaLOSMT / magDeltaLOSMT;
%Lateral Acceleration Unit Vector
            magncM = navCoefM * magLOSRateMT * VcMT;
%Magnitude of Lateral Acceleration (m/s^2)

            ncM = magncM * unitncM;
%Lateral Acceleration Vector (m/s^2)

            %Reset counters/flags
            txCounter = 0;
            txFlag = 0;
            oldDistMT = distMT;
            oldLOSMT = LOSMT;
        end %txFlag | (txCounter >= updateTime)

        %Compute Target Acceleration Perpendicular to LOS (Target Maneuver)
        magaT = sqrt(aT(1) ^ 2 + aT(2) ^ 2 + aT(3) ^ 2);
%Magnitude of Target Acceleration (m/s^2)
        unitaT = aT / magaT;
%Target Acceleration Unit Vector
        alfa = acos(dot(unitaT, -unitLOSMT));
%Angle Between Target Acceleration Vector and LOS (rad)
        magaPLOST = magaT * sin(alfa);
%Target Acceleration Component Perpendicular to LOS (m/s^2)
        mnvrT = magaPLOST / gT;
%Target Maneuver (g)

        if TMc == 0 %Control System Dynamics Implementation
            nlM = ncM;
        else
            %Implement Control System Dynamics
            %x-axis
            stateMNextX = AMc * stateMX + BMc * ncM(1);
            nlMX = CMc * stateMX + DMc * ncM(1);
            stateMX = stateMNextX;
            %y-axis
            stateMNextY = AMc * stateMY + BMc * ncM(2);
            nlMY = CMc * stateMY + DMc * ncM(2);
            stateMY = stateMNextY;
            %z-axis
            stateMNextZ = AMc * stateMZ + BMc * ncM(3);
            nlMZ = CMc * stateMZ + DMc * ncM(3);
            stateMZ = stateMNextZ;

            nlM = [nlMX(1); nlMY(1); nlMZ(1)];  %Achieved lateral acceleration vector
            if nlM == [0;0;0]
                nlM = ncM;
            end
        end %TMc == 0 (Control System Dynamics Implementation)

        magnlM = sqrt(nlM(1) ^ 2 + nlM(2) ^ 2 + nlM(3) ^ 2);
%Magnitude of achieved lateral acceleration

        comLatAccM = magncM / gM;               %Commanded Lateral Acceleration (g)
        achLatAccM = magnlM / gM;               %Achieved Lateral Acceleration (g)

        latDivM = latDivM + abs(magnlM * dt);   %Lateral Divert (m/s)

        %Compute Lateral Acceleration Perpendicular to Velocity Vector, Ignore Parallel Component
        unitnlM = nlM / magnlM;                 %Unit Achieved Acceleration
        beta = acos(dot(unitnlM, unitvM));
%Angle between Velocity Vector and Achieved Acceleration Vector (rad)
        magnlParM = magnlM * cos(beta);
%Magnitude of Achieved Acceleration Parallel to Velocity Vector (Ignore This)
        nlParM = unitvM * magnlParM;
%Achieved Acceleration Vector Parallel to Velocity Vector (Ignore This)
        nlPerM = nlM - nlParM;
%Achieved Acceleration Vector Perpendicular to Velocity Vector (Use This)
        magnlPerM = sqrt(nlPerM(1) ^ 2 + nlPerM(2) ^ 2 + nlPerM(3) ^ 2);
%Magnitude of Acceleration Vector Perpendicular to Velocity Vector
        unitnlPerM = nlPerM / magnlPerM;
%Unit achieved Acceleration Vector Perpendicular to Velocity Vector

        GFM = nlPerM * MM;                      %Guidance Force (N) (Perpendicular to LOS)

        magGFM= sqrt(GFM(1) ^ 2 + GFM(2) ^ 2 + GFM(3) ^ 2); %Magnitude of guidance force
        unitGFM = GFM / magGFM;                 %Unit guidance force vector
    end

    oldDistMTTrue = distMTTrue; %Record old Missile-target distance

    %Record Data
    tArray = [tArray t];                                    %Simulation time
    %Target
    posArrayT = [posArrayT posT];                           %Target position
    sensedPosArrayT = [sensedPosArrayT sensedPosT];         %Sensed target position
    hArrayT = [hArrayT hT];                                 %Target height
    groundTrackArrayT = [groundTrackArrayT groundTrackT];   %Target ground track
    distArrayT = [distArrayT distT];                        %Target downrange
    vArrayT = [vArrayT magvT];                              %Target velocity
    stageArrayT = [stageArrayT stageT];                     %Target stage
    massArrayT = [massArrayT MT];                           %TArget mass
    mnvrArrayT = [mnvrArrayT mnvrT];                        %Target maneuver
    %Missile
    posArrayM = [posArrayM posM];                         %Missile position
    hArrayM = [hArrayM hM];                               %Missile height
    groundTrackArrayM = [groundTrackArrayM groundTrackM]; %Missile groundtrack
    distArrayM = [distArrayM distM];                      %Missile downrange
    vArrayM = [vArrayM magvM];                            %Missile velocity
    stageArrayM = [stageArrayM stageM];                   %Missile stage
    massArrayM = [massArrayM MM];                         %Missile Mass
    comLatAccArrayM = [comLatAccArrayM comLatAccM];       %Missile commanded lateral acceleration
    achLatAccArrayM = [achLatAccArrayM achLatAccM];       %Missile achieved lateral acceleration
    latDivArrayM = [latDivArrayM latDivM];                %Missile Lateral divert
    %Decoy
    posArrayD = [posArrayD posD];                         %Decoy position
    sensedPosArrayD = [sensedPosArrayD sensedPosD];       %Sensed decoy position
    hArrayD = [hArrayD hD];                               %Decoy height
    groundTrackArrayD = [groundTrackArrayD groundTrackD]; %Decoy groundtrack
    distArrayD = [distArrayD distD];                      %Decoy downrange
    vArrayD = [vArrayD magvD];                            %Decoy velocity
    %Target-Decoy
    distTDArray = [distTDArray distTD];                   %Target-Decoy distance
    %Missile-Target
    distArrayMT = [distArrayMT distMT];                   %Missile-target distance
    VcArrayMT = [VcArrayMT VcMT];                         %Missile-target closure velocity
    %Sensor
    distRF1TArray = [distRF1TArray distRF1T];             %RF-1-Target distance
    distRF2TArray = [distRF2TArray distRF2T];             %RF-2-Target distance
    RCS1Array = [RCS1Array (10*log10(RCS1))];             %RCS Seen by RF-1 (dBsm)
    RCS2Array = [RCS2Array (10*log10(RCS2))];             %RCS Seen by RF-2 (dBsm)
    SNR1Array = [SNR1Array SNR1dB];                       %SNR seen by RF-1
    SNR2Array = [SNR2Array SNR2dB];                       %SNR seen by RF-2
    SJR1Array = [SJR1Array SJR1dB];                       %SJR seen by RF-1
    SJR2Array = [SJR2Array SJR2dB];                       %SJR seen by RF-2
    sigmaAlfaRF1Array = [sigmaAlfaRF1Array sigmaAlfaRF1]; %RMS angular error introduced by RF-1
    sigmaAlfaRF2Array = [sigmaAlfaRF2Array sigmaAlfaRF2]; %RMS angular error introduced by RF-2
    sigmaAlfaRRF1Array = [sigmaAlfaRRF1Array sigmaAlfaRRF1];
%RMS angular error at target range introduced by RF-1
    sigmaAlfaRRF2Array = [sigmaAlfaRRF2Array sigmaAlfaRRF2];
%RMS angular error at target range introduced by RF-2
    sigmaRRF1Array = [sigmaRRF1Array sigmaRRF1];          %RMS range error introduced by RF-1
    sigmaRRF2Array = [sigmaRRF2Array sigmaRRF2];          %RMS range error introduced by RF-2
    magErrPosTRF1Array = [magErrPosTRF1Array magErrPosTRF1];
%Magnitude of position error introduced by RF-1
    magErrPosTRF2Array = [magErrPosTRF2Array magErrPosTRF2];
%Magnitude of position error introduced by RF-2
    magErrPosTIRArray = [magErrPosTIRArray magErrPosTIR];
%Magnitude of position error introduced by IR
    magPosErrorArray = [magPosErrorArray magPosError];    %Magnitude of position error
    JTArray = [JTArray JT];                               %Target radiation intensity
    SCRIR1dBArray = [SCRIR1dBArray SCRIR1dB];             %Signal/Clutter ratio IR-1
    SCRIR2dBArray = [SCRIR2dBArray SCRIR2dB];             %Signal/Clutter ratio IR-2
    lookRF1Array = [lookRF1Array (lookRF1*180/pi)];       %Target aspect seen by RF-1
    lookRF2Array = [lookRF2Array (lookRF2*180/pi)];       %Target aspect seen by RF-2

    t = t + dt;                             %Increase Time
    txCounter = txCounter + dt;             %Update Time Counter

    if t >= 3600                            %Exit After 1 Hour Anyway
        break;
    end
end %((hT >= SMALL) | (hM >= SMALL)) & ((VcMTTrue >= 0) | (t < 90)) (Main loop)

%Erase Data After Miss
passIndex = find(distArrayMT == min(distArrayMT))-1;
%Target
tArray = tArray(1:size(tArray,1), 1:passIndex);
posArrayT = posArrayT(1:size(posArrayT ,1), 1:passIndex);
sensedPosArrayT = sensedPosArrayT(1:size(sensedPosArrayT ,1), 1:passIndex);
hArrayT = hArrayT(1:size(hArrayT ,1), 1:passIndex);
groundTrackArrayT = groundTrackArrayT(1:size(groundTrackArrayT ,1), 1:passIndex);
distArrayT = distArrayT(1:size(distArrayT ,1), 1:passIndex);
vArrayT = vArrayT(1:size(vArrayT ,1), 1:passIndex);
stageArrayT = stageArrayT(1:size(stageArrayT ,1), 1:passIndex);
massArrayT = massArrayT(1:size(massArrayT ,1), 1:passIndex);
mnvrArrayT = mnvrArrayT(1:size(mnvrArrayT ,1), 1:passIndex);
%Missile
posArrayM = posArrayM(1:size(posArrayM ,1), 1:passIndex);
hArrayM = hArrayM(1:size(hArrayM ,1), 1:passIndex);
groundTrackArrayM = groundTrackArrayM(1:size(groundTrackArrayM ,1), 1:passIndex);
distArrayM = distArrayM(1:size(distArrayM ,1), 1:passIndex);
vArrayM = vArrayM(1:size(vArrayM ,1), 1:passIndex);
stageArrayM = stageArrayM(1:size(stageArrayM ,1), 1:passIndex);
```

```
massArrayM = massArrayM(1:size(massArrayM ,1), 1:passIndex);
comLatAccArrayM = comLatAccArrayM(1:size(comLatAccArrayM ,1), 1:passIndex);
achLatAccArrayM = achLatAccArrayM(1:size(achLatAccArrayM ,1), 1:passIndex);
latDivArrayM = latDivArrayM(1:size(latDivArrayM ,1), 1:passIndex);
%Decoy
posArrayD = posArrayD(1:size(posArrayD ,1), 1:passIndex);
sensedPosArrayD = sensedPosArrayD(1:size(sensedPosArrayD ,1), 1:passIndex);
hArrayD = hArrayD(1:size(hArrayD ,1), 1:passIndex);
groundTrackArrayD = groundTrackArrayD(1:size(groundTrackArrayD ,1), 1:passIndex);
distArrayD = distArrayD(1:size(distArrayD ,1), 1:passIndex);
vArrayD = vArrayD(1:size(vArrayD ,1), 1:passIndex);
%Target-Decoy
distTDArray = distTDArray(1:size(distTDArray ,1), 1:passIndex);
%Missile-Target
distArrayMT = distArrayMT(1:size(distArrayMT ,1), 1:passIndex);
VcArrayMT = VcArrayMT(1:size(VcArrayMT, 1), 1:passIndex);
%Sensor
distRF1TArray = distRF1TArray(1:size(distRF1TArray ,1), 1:passIndex);
distRF2TArray = distRF2TArray(1:size(distRF2TArray ,1), 1:passIndex);
RCS1Array = RCS1Array(1:size(RCS1Array ,1), 1:passIndex);
RCS2Array = RCS2Array(1:size(RCS2Array ,1), 1:passIndex);
SNR1Array = SNR1Array(1:size(SNR1Array ,1), 1:passIndex);
SNR2Array = SNR2Array(1:size(SNR2Array ,1), 1:passIndex);
SJR1Array = SJR1Array(1:size(SJR1Array ,1), 1:passIndex);
SJR2Array = SJR2Array(1:size(SJR2Array ,1), 1:passIndex);
sigmaAlfaRF1Array = sigmaAlfaRF1Array(1:size(sigmaAlfaRF1Array ,1), 1:passIndex);
sigmaAlfaRF2Array = sigmaAlfaRF2Array(1:size(sigmaAlfaRF2Array ,1), 1:passIndex);
sigmaAlfaRRF1Array = sigmaAlfaRRF1Array(1:size(sigmaAlfaRRF1Array ,1), 1:passIndex);
sigmaAlfaRRF2Array = sigmaAlfaRRF2Array(1:size(sigmaAlfaRRF2Array ,1), 1:passIndex);
sigmaRRF1Array = sigmaRRF1Array(1:size(sigmaRRF1Array ,1), 1:passIndex);
sigmaRRF2Array = sigmaRRF2Array(1:size(sigmaRRF2Array ,1), 1:passIndex);
magErrPosTRF1Array = magErrPosTRF1Array(1:size(magErrPosTRF1Array ,1), 1:passIndex);
magErrPosTRF2Array = magErrPosTRF2Array(1:size(magErrPosTRF2Array ,1), 1:passIndex);
magErrPosTIRArray = magErrPosTIRArray(1:size(magErrPosTIRArray ,1), 1:passIndex);
magPosErrorArray = magPosErrorArray(1:size(magPosErrorArray ,1), 1:passIndex);
JTArray = JTArray(1:size(JTArray ,1), 1:passIndex);
SCRIR1dBArray = SCRIR1dBArray(1:size(SCRIR1dBArray ,1), 1:passIndex);
SCRIR2dBArray = SCRIR2dBArray(1:size(SCRIR2dBArray ,1), 1:passIndex);

%Define Earth
[xE, yE, zE] = sphere(36);                              %Generate Unit Sphere
xE = xE .* Re;                                          %Expand X-axis
yE = yE .* Re;                                          %Expand Y-axis
zE = zE .* Re;                                          %Expand Z-axis

%Visualize Earth & The Coordinate Frame
figure(3);
axis equal;
axis([-7e6 7e6 -7e6 7e6 -7e6 7e6]);                     %Set Axes
view(280,30);                                           %Set Suitable View
grid on;
hold on;
surf(xE, yE, zE);                                       %Plot Earth

%3D Target Trajectory and GroundTrack
title('Trajectories & Groundtracks')
xlabel('x(m)');
ylabel('y(m)');
zlabel('z(m)');

%Plot Target Trajectory
posArrayTx = posArrayT(1,:);
posArrayTy = posArrayT(2,:);
posArrayTz = posArrayT(3,:);
plot3(posArrayTx, posArrayTy, posArrayTz, 'y-');

%Plot Missile Trajectory
posArrayMx = posArrayM(1,:);
posArrayMy = posArrayM(2,:);
posArrayMz = posArrayM(3,:);
plot3(posArrayMx, posArrayMy, posArrayMz, 'b--');

%Plot Decoy Trajectory
posArrayDx = posArrayD(1,:);
posArrayDy = posArrayD(2,:);
posArrayDz = posArrayD(3,:);
plot3(posArrayDx, posArrayDy, posArrayDz, 'g-.');

%Plot Target Groundtrack
groundTrackArrayTx = groundTrackArrayT(1,:);
groundTrackArrayTy = groundTrackArrayT(2,:);
```

```
groundTrackArrayTz = groundTrackArrayT(3,:);
plot3(groundTrackArrayTx, groundTrackArrayTy, groundTrackArrayTz, 'k:');

%Plot Missile Groundtrack
groundTrackArrayMx = groundTrackArrayM(1,:);
groundTrackArrayMy = groundTrackArrayM(2,:);
groundTrackArrayMz = groundTrackArrayM(3,:);
plot3(groundTrackArrayMx, groundTrackArrayMy, groundTrackArrayMz, 'k:');

%Plot Decoy Groundtrack
groundTrackArrayDx = groundTrackArrayD(1,:);
groundTrackArrayDy = groundTrackArrayD(2,:);
groundTrackArrayDz = groundTrackArrayD(3,:);
plot3(groundTrackArrayDx, groundTrackArrayDy, groundTrackArrayDz, 'k:');

plot3(pos0T(1), pos0T(2), pos0T(3), 'bo')
plot3(pos0M(1), pos0M(2), pos0M(3), 'go')
plot3(posFA(1), posFA(2), posFA(3), 'ro')
plot3(posRF1(1), posRF1(2), posRF1(3), 'co');
plot3(posRF2(1), posRF2(2), posRF2(3), 'mo');
plot3(posIR1(1), posIR1(2), posIR1(3), 'ko');
plot3(posIR2(1), posIR2(2), posIR2(3), 'ko');

legend('Target Trajectory','Missile Trajectory','Decoy Trajectory'); %,'','Target Launch
Site','Missile Launch Site','Friendly Asset');

%Plot Distance vs Height
figure(4);
hold on;
plot((distArrayT ./ 1000), (hArrayT ./ 1000),'r-');
plot((distArrayM ./ 1000), (hArrayM ./ 1000),'b--');
plot((distArrayD ./ 1000), (hArrayD ./ 1000),'g-.');
title('Ground Distance vs. Height');
xlabel('Ground Distance (km)');
ylabel('Height (km)');
legend('Target','Missile', 'Decoy');

%Plot Speed vs Time
figure(5);
hold on;
plot((tArray ./ 60), (vArrayT ./ 1000),'r-');
plot((tArray ./ 60), (vArrayM ./ 1000),'b--');
plot((tArray ./ 60), (vArrayD ./ 1000),'g-.');
title('Velocity vs. Flight Time');
xlabel('Flight Time (min)');
ylabel('Velocity (km/s)');
legend('Target','Missile','Decoy');

%Plot Stage vs Time
figure(6);
hold on;
plot((tArray ./ 60), stageArrayT,'r');
plot((tArray ./ 60), stageArrayM,'b');
title('Stage vs. Flight Time');
xlabel('Flight Time (min)');
ylabel('Stage');
legend('Target','Missile');

%Plot Total Mass vs Time
figure(7);
hold on;
plot((tArray ./ 60), massArrayT ./ 1000,'r');
plot((tArray ./ 60), massArrayM ./ 1000,'b');
title('Total Mass vs. Flight Time');
xlabel('Flight Time (min)');
ylabel('Mass (Tons)');
legend('Target','Missile');

%Plot Missile-Target Distance
figure(8);
plot((tArray ./ 60), distArrayMT ./ 1000,'b');
title('Missile-Target Distance');
xlabel('Time (min)');
ylabel('Distance (km)');

%Plot Missile-Target Closure Velocity
figure(9);
plot((tArray ./ 60), VcArrayMT ./ 1000,'b');
axis([0 (max(tArray) /60) 0 11]);
title('Missile-Target Closure Velocity');
xlabel('Time (min)');
```

```
ylabel('Vc (km/s)');

%Plot Missile Lateral Acceleration
figure(10);
plot((tArray ./ 60), comLatAccArrayM ,'b');
hold on
plot((tArray ./ 60), achLatAccArrayM ,'r');
title('Missile Lateral Acceleration');
xlabel('Time (min)');
ylabel('Lateral Acceleration (g)');
axis([0 (max(tArray)/60) 0 max(achLatAccArrayM)]);
legend('Commanded', 'Achieved');

%Plot Missile Lateral Divert
figure(11);
plot((tArray ./ 60), latDivArrayM ,'b');
title('Missile Lateral Divert');
xlabel('Time (min)');
ylabel('Lateral Divert (m/s)');

%Plot Target Maneuver
figure(12);
plot((tArray ./ 60), mnvrArrayT ,'b');
title('Target Maneuver');
xlabel('Time (min)');
ylabel('Maneuver (g)');

%Plot RCS Seen by RF-1
figure(13);
plot((tArray ./ 60), RCS1Array ,'b-');
hold on
plot((tArray ./ 60), RCS2Array ,'r--');
title('RCS Seen by RF-1 and RF-2');
xlabel('Time (min)');
ylabel('RCS (dBsm)');
legend('RF-1', 'RF-2');

%Plot RF Sensor Target Distance
figure(14);
plot((tArray ./ 60), (distRF1TArray ./ 1000) ,'b-');
hold on
plot((tArray ./ 60), (distRF2TArray ./ 1000) ,'r--');
title('RF Sensor-Target Distance');
xlabel('Time (min)');
ylabel('Distance (km)');
legend('RF-1', 'RF-2');

%RF Sensor SNR
figure(15);
plot((tArray ./ 60), (SNR1Array) ,'b-');
hold on
plot((tArray ./ 60), (SNR2Array) ,'r--');
title('RF Sensor SNR (Single Pulse)');
xlabel('Time (min)');
ylabel('SNR (dB)');
legend('RF-1', 'RF-2');

%RF Sensor SJR
figure(16);
plot((tArray ./ 60), (SJR1Array) ,'b-');
hold on
plot((tArray ./ 60), (SJR2Array) ,'r--');
title('RF Sensor SJR (Single Pulse)');
xlabel('Time (min)');
ylabel('SJR (dB)');
legend('RF-1', 'RF-2');


%RF Sensor Range Accuracy
figure(17);
plot((tArray ./ 60), (sigmaRRF1Array) ,'b-');
hold on
plot((tArray ./ 60), (sigmaRRF2Array) ,'r--');
title('RMS Error in Range');
xlabel('Time (min)');
ylabel('RMS Error in Range (m)');
legend('RF-1', 'RF-2');

%RF Sensor Angle Accuracy
figure(18);
plot((tArray ./ 60), (sigmaAlfaRF1Array .* 180 ./ pi) ,'b-');
```

```
hold on
plot((tArray ./ 60), (sigmaAlfaRF2Array .* 180 ./ pi) ,'r--');
title('RMS Error in Angle');
xlabel('Time (min)');
ylabel('RMS Error in Angle (deg)');
legend('RF-1', 'RF-2');

%RF Sensor Angle Accuracy
figure(19);
plot((tArray ./ 60), sigmaAlfaRRF1Array ,'b-');
hold on
plot((tArray ./ 60), sigmaAlfaRRF2Array ,'r--');
title('RMS Angular Error at Target Range');
xlabel('Time (min)');
ylabel('RMS Angular Error at Target Range (m)');
legend('RF-1', 'RF-2');

%Position Error Introduced by RF-1
figure(20);
plot((tArray ./ 60), magErrPosTRF1Array ,'b-');
title('Position Error Introduced by RF-1');
xlabel('Time (min)');

ylabel('Magnitude of Position Error (m)');

%Position Error Introduced by RF-2
figure(21);
plot((tArray ./ 60), magErrPosTRF2Array ,'b-');
title('Position Error Introduced by RF-2');
xlabel('Time (min)');
ylabel('Magnitude of Position Error (m)');

%Position Error Introduced by IR
figure(22);
plot((tArray ./ 60), magErrPosTIRArray ,'b-');
title('Position Error Introduced by IR Sensors');
xlabel('Time (min)');
ylabel('Magnitude of Position Error (m)');

%Position Error
figure(23);
plot((tArray ./ 60), magPosErrorArray ,'b-');
title('Position Error');
xlabel('Time (min)');
ylabel('Magnitude of Position Error (m)');

%Radiation Intensity
figure(24);
plot((tArray ./ 60), (JTArray ./ 1e6) ,'b-');
title('Radiation Intesity versus Time');
xlabel('Time (min)');
ylabel('Radiation Intensity (MW/sr)');

%Signal-to-Clutter Ratio
figure(25);
plot((tArray ./ 60), SCRIR1dBArray ,'b-');
hold on
plot((tArray ./ 60), SCRIR2dBArray ,'r--');
legend('IR-1','IR-2')
title('S/C Ratio versus Time');
xlabel('Time (min)');
ylabel('Signal-to-Clutterb Ratio (dB)');

%Target-Decoy Distance
figure(26)
plot((tArray ./ 60), (distTDArray./1000) ,'b-');
title('Target-Decoy Separation');
xlabel('Time (min)');
ylabel('Target-Decoy Distance (km)');

%Display Final Intercept Data
disp (['Target Range =' num2str(distT / 1000) ' km.'])
disp (['Missile Range =' num2str(distM / 1000) ' km.'])
disp (' ' );
disp (['Intercept Time =' num2str(t/60) ' minutes.'])
disp (['Miss Distance =' num2str(min(distArrayMT)) ' m.'])
disp (['Lateral Divert =' num2str(max(latDivArrayM)) ' m/s.'])
disp (' ');
disp ('Simulation Finished.');
```

LIST OF REFERENCES

1. Charles A. Fowler, "National missile defense," *IEEE Aerospace and Electronics Systems Magazine,* Vol. 17, No. 1, pp. 4-12, 2002.

2. David E. Mosher, "The grand plans," *IEEE Spectrum,* Vol. 34, No. 9, pp. 28-39, 1997.

3. Dov S. Zakheim, "Old rivalries, new arsenals: should the United States worry?," *IEEE Spectrum,* Vol. 36, No. 3, pp. 29-31, 1999.

4. James Oberg, "Missiles for all: the new global threat," *IEEE Spectrum,* Vol. 36, No. 3, pp. 20-28, 1999.

5. Jennifer C. Davis and James J. Lisowski, "Developing a remote staring sensor for optimizing successful boost phase intercept," *Proc. of IEEE Aerospace Conference,* pp. 4-1649–4-1661, IEEE Press, Piscataway, New Jersey, Cat. No. 02TH8593, 2002.

6. George N. Lewis and Theodore A. Postol, "Future challenges to ballistic missile defense," *IEEE Spectrum,* Vol. 34, No. 9, pp. 60-68, 1997.

7. Andrew M. Sessler, John M. Cornwall, Bob Dietz, Steve Fetter, Sherman Frankel, Richard L. Garwin, et al., "Countermeasures: A technical evaluation of the operational effectiveness of the planned US national missile defense system," Union of Concerned Scientists, MIT Security Studies Program, Cambridge, Massachusetts, April 2000.

8. Paul Zarchan, *Tactical and Strategic Missile Guidance*, American Institute of Aeronautics and Astronautics, Reston, Virginia, 2002.

9. Dennis Roddy, *Satellite Communications*, McGraw-Hill, New York, 2001.

10. David Halliday, Robert Resnick and Jearl Walker, *Fundamentals of Physics*, John Wiley & Sons, New York, 1997.

11. Federation of American Scientists, "LGM-118A Peacekeeper", http://www.fas.org/nuke/guide/usa/icbm/lgm-118.htm, last accessed 13 September 2004.

12. Hangju Cho, Chang Kyung Ryoo and Min-Jea Tahk, "Implementation of optimal guidance laws using predicted missile velocity profiles," *J. of Guidance, Control and Dynamics*, Vol. 22, No. 4, pp. 579-588, 1999.

13. Pini Gurfil, Mario Jodorkovsky and Moshe Guelman, "Design of nonsaturating guidance systems," *J. of Guidance, Control and Dynamics*, Vol. 23, No. 4, pp. 693-700, 2000.

14. E. Garrido, “Graphical user interface for a physical optics radar cross section prediction code,” Master’s Thesis, Naval Postgraduate School, Monterey, California, 2000.

15. Donald W. Wilmot, William R. Owens and Robert J. Shelton, “Warning systems,” in *Electro-optical Systems Handbook Vol. 7: Countermeasure Systems*, David H. Pollock, ed., Environmental Research Institute of Michigan, Ann Arbor, Michigan and SPIE Optical Engineering Press, Bellingham, Washington, 1996.

16. Filippo Neri, *Introduction to Electronic Defense Systems*, Artech House, Norwood, Massachusetts, 2001.

17. Merrill I. Skolnik, *Introduction to Radar Systems*, McGraw Hill, New York, 2001.

18. Warren L. Stutzman and Gary A. Thiele, *Antenna Theory and Design*, John Wiley and Sons, New York, 1998.

19. Alfred Cooper, Notes for PH3204 *(Electro-optic Systems and Countermeasures)*, Naval Postgraduate School, 2003 (unpublished).

20. Phillip E. Pace, Notes for EC3700 *(Introduction to Joint Services Electronic Warfare)*, Naval Postgraduate School, 2004 (unpublished).

21. Sergei A. Vakin, Lev N. Shustov, and Robert H. Dunwell, *Fundamentals of Electronic Warfare*, Artech House, Norwood, Massachusetts, 2001.

22. Gokhan Humali, “Sensor fusion for boost phase interception of ballistic missiles,” Master’s Thesis, Naval Postgraduate School, Monterey, California, 2004.

23 John Bardanis, “Kill vehicle effectiveness for boost phase interception of ballistic missiles,” Master’s Thesis, Naval Postgraduate School, Monterey, California, 2004.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. John Powers
   Department of Electrical and Computer Engineering
   Monterey, California

4. Dr. Dan Boger
   Information Sciences Department
   Monterey, California

5. Dr. Phillip E. Pace
   Department of Electrical and Computer Engineering
   Monterey, California

6. Dr. Murali Tummala
   Department of Electrical and Computer Engineering
   Monterey, California

7. Captain Kubilay Uzun
   Turkish Air Force
   Ankara, Turkey

8. Mr. Dale S. Caffall
   Missile Defense Agency
   Washington, D.C.

9. Ms. Gerri Hudson
   Raytheon Company
   Tucson, Arizona

10. Mr. Howard Krizek
    Raytheon Company
    Tucson, Arizona

11. Mr. Lamoyne Taylor
    Raytheon Company
    Tucson, Arizona